НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004:004.453

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___"_____2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 122 – комп'ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва спеціальності)

на тему: Емоційна нейронна діалогова система зі знаннями для пошуку онлайн курсів

Виконав (-ла): студент (-ка) 6 курсу, групи ДА-62м
(шифр групи)

Шаптала Роман Віталійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н. Кисельов Г. Д. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту к.т.н. Кисельов Г. Д. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент проф., д.т.н. Аушева Н.М. _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 122 – комп'ютерні науки та інформаційні технології (Системне проектування сервісів)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Шапталі Роману Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Емоційна нейронна діалогова система зі знаннями для пошуку онлайн курсів
науковий керівник дисертації К.т.н. Кисельов Геннадій Дмитрович ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження процес текстової взаємодії людина-машина
_____

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) діалогова система з емоціями та знаннями на тему онлайн курсів і здатністю рекомендувати їх
_____
_____

5. Перелік завдань, які потрібно розробити
1. Емоційна нейронна діалогова система зі знаннями.

2. Розробка емоційної нейронної діалогової системи зі знаннями.
3. Огляд результатів розробки системи.
4. Розробка стартап-проекту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу _____
презентація на тему “Емоційна нейронна діалогова система зі знаннями для пошуку онлайн курсів”

7. Орієнтовний перелік публікацій

1. Shaptala R. Exploring the vector space model for online courses / R. Shaptala, A. Kyselova, G. Kyselov. // 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON). – 2017. – С. 861–864.
2. Shaptala R. Evaluation of approaches to emotion classification / R. Shaptala, A. Kyselova, G. Kyselov. // System analysis and information technology: 20-th International conference SAIT 2018, Kyiv, Ukraine.
3. Shaptala R. Neural dialogue system with emotion embeddings / R. Shaptala, A. Kyselova, G. Kyselov. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка стартап-проекту	Кисельов Г. Д., доцент		

9. Дата видачі завдання 01.02.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	2	3	4
1	Отримання завдання	01.02.2018	
2	Побудова класифікатора емоцій у текстах	15.02.2018	
3	Побудова базової seq2seq моделі	05.03.2018	
4	Побудова емоційної діалогової системи	30.03.2018	
5	Створення модулю знань для діалогової системи	10.04.2018	
6	Розробка інтерфейсу спілкування з діалоговою системою	17.04.2018	

1	2	3	4
7	Оцінка результатів тренування системи	20.04.2018	
8	Отримання допуску до захисту та подача роботи в ДЕК	10.05.2018	

Студент _____ Р.В. Шаптала
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Г.Д. Кисельов
(підпис) (ініціали, прізвище)

РЕФЕРАТ

магістерської дисертації Шаптали Романа Віталійовича на тему

«Емоційна нейронна діалогова система зі знаннями для пошуку онлайн курсів»

Робота виконана на 82 сторінках, містить 9 ілюстрацій, 26 таблиць. При підготовці використовувалась література з 38 джерел.

Актуальність теми дослідження полягає в тому, що поточні діалогові системи охоплюють лише семантичну та граматичну складові природньої мови, ігноруючи емоційне забарвлення необхідне при живому спілкуванні, а тому вимагають механізмів інтеграції емоційності при розмові. Крім того, здатність надати таким системам специфічних знань та можливості здійснювати обдуманий пошук та рекомендації є дуже важким завданням та потребує подальших досліджень. Таким чином створення діалогової системи, яка може імпонувати емоціям користувача та має знання про онлайн курси, здатне сприяти прискореному навчанню фахівців у певній області.

Мета та задачі дослідження. Метою даної роботи є створення емоційної нейронної діалогової системи зі знаннями для пошуку онлайн курсів. Поставлена мета вимагає вирішення наступних наукових задач:

1) аналіз та класифікація емоцій у текстових даних, зокрема діалогових репліках;
2) розробка механізму емоційності у нейронних діалогових системах;
3) розробка семантичної моделі пошуку онлайн курсів за їх текстовим описом.

Об’єкт досліджень - процес текстової взаємодії людина-машина.

Предмет досліджень - діалогова система з емоціями та знаннями на тему онлайн курсів і здатністю рекомендувати їх.

Методи досліджень. Для вирішення проблеми в даній роботі використовуються методи аналізу і синтезу, системного аналізу, порівняння, логічного узагальнення результатів, проектування логічних структур даних.

Наукова новизна. Наукова новизна дипломної роботи полягає в тому, що розроблена система є першою емоційною нейронною діалоговою системою для рекомендації онлайн курсів. У ній при спілкуванні на сторонні від освіти теми, можна контролювати емоційність відповідей даної системи, що не було можливим у попередніх розробках, пов’язаних з НДС. Потенційні застосування та практична цінність результатів дипломної роботи:

1) Розроблений механізм емоційності можна застосувати для покращення існуючих нейронних діалогових систем та додатків на їх основі;

2) Розроблений метод контролю емоційного забарвлення відповідей можна розширити і для контролю інших аспектів природнього спілкування, наприклад стилю мовлення;

3) Дану систему можна запровадити як додатковий ресурс пошуку матеріалів у освітніх системах.

Апробації результатів дисертації. Результати досліджень оприлюднені на 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON).

Публікації:

1. Shaptala R. Exploring the vector space model for online courses / R. Shaptala, A. Kyselova, G. Kyselov. // 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON). – 2017. – С. 861–864.
2. Прийнято: Shaptala R. Evaluation of approaches to emotion classification / R. Shaptala, A. Kyselova, G. Kyselov. // System analysis and information technology: 20-th International conference SAIT 2018, Kyiv, Ukraine.
3. Подано: Shaptala R. Neural dialogue system with emotion embeddings / R. Shaptala, A. Kyselova, G. Kyselov. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

Ключові слова: нейронна діалогова система, емоційність, рекурентна нейронна мережа, FastText, вектор сутності.

РЕФЕРАТ

магистерской диссертации Шапталы Романа Витальевича на тему «Эмоциональная нейронная диалоговая система со знаниями для поиска онлайн курсов»

Работа выполнена на 82 страницах, содержит 9 иллюстраций, 26 таблиц. При подготовке использовалась литература из 38 источников.

Актуальность темы исследования заключается в том, что текущие диалоговые системы охватывают только семантическую и грамматическую составляющие естественного языка, игнорируя эмоциональную окраску необходимую при живом общении, а потому требуют механизмов интеграции эмоциональности при разговоре. Кроме того, способность предоставить таким системам специфических знаний и возможности осуществлять обдуманный поиск и рекомендации является очень трудной задачей и требует дальнейших исследований. Таким образом создание диалоговой системы, которая может импонировать эмоциям пользователя и имеет знания об онлайн курсах, способно содействовать ускоренному обучению специалистов в определенной области.

Цель и задачи исследования. Целью данной работы является создание эмоциональной нейронной диалоговой системы со знаниями для поиска онлайн курсов. Поставленная цель требует решения следующих научных задач:

1) анализ и классификация эмоций в текстовых данных, в частности диалоговых репликах;

2) разработка механизма эмоциональности в нейронных диалоговых системах;

3) разработка семантической модели поиска онлайн курсов по их текстовым описаниям.

Объект исследований - процесс текстового взаимодействия человек-машина.

Предмет исследований - диалоговая система с эмоциями и знаниями на тему онлайн курсов и способностью рекомендовать их.

Методы исследования. Для решения проблемы в данной работе используются методы анализа и синтеза, системного анализа, сравнения,

логического обобщения результатов, проектирования логических структур данных.

Научная новизна. Научная новизна дипломной работы заключается в том, что разработанная система является первой эмоциональной нейронной диалоговой системой для рекомендации онлайн курсов. В ней при общении на посторонние от образования темы, можно контролировать эмоциональность ответов данной системы, что не было возможным в предыдущих разработках, связанных с НДС. Потенциальные применения и практическая ценность результатов дипломной работы:

1) Разработанный механизм эмоциональности можно применить для улучшения существующих нейронных диалоговых систем и приложений на их основе;

2) Разработанный метод контроля эмоциональной окраски ответов можно расширить и для контроля других аспектов естественного общения, например стиля речи;

3) Данную систему можно ввести как дополнительный ресурс поиска материалов в образовательных системах.

Аппробация результатов диссертации. Результаты исследования опубликованы на 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON).

Публикации:

1. Shaptala R. Exploring the vector space model for online courses / R. Shaptala, A. Kyselova, G. Kyselov. // 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON). – 2017. – С. 861–864.
2. Прийнято: Shaptala R. Evaluation of approaches to emotion classification / R. Shaptala, A. Kyselova, G. Kyselov. // System analysis and information technology: 20-th International conference SAIT 2018, Kyiv, Ukraine.
3. Подано: Shaptala R. Neural dialogue system with emotion embeddings / R. Shaptala, A. Kyselova, G. Kyselov. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

Ключевые слова: нейронная диалоговая система, эмоциональность, рекуррентная нейронная сеть, FastText, вектор сущности.

ABSTRACT

for master's thesis by Roman Shaptala on " Emotional neural dialogue system with knowledge for online courses search "

Work carried out on 82 pages containing 9 figures and 26 tables. The paper was written with references to 38 different sources.

The relevance of the research topic lies in the fact that the current dialogue systems cover only the semantic and grammatical components of the natural language, ignoring the emotional coloring necessary for live communication, and therefore require mechanisms for integrating emotionality in conversation. In addition, the ability to provide such systems with specific knowledge and the ability to carry out thoughtful queries and recommendations is a very difficult task and requires further research. Thus, the creation of an interactive system that can take into account the user's emotions and has knowledge of online courses, can promote accelerated training of specialists in a certain field.

Purpose and objectives of the study. The purpose of this work is to create an emotional neural dialogue system with knowledge for online courses search. The goal is to solve the following scientific problems:

1) analysis and classification of emotions in text data, in particular, dialog utterances;

2) development of the mechanism of emotionality in neural dialogue systems;

3) development of a semantic model for searching online courses via their text descriptions.

The object of research is the process of textual human-machine interaction.

The subject of research is an interactive system with emotions and knowledge on the topic of online courses and the ability to recommend them.

To solve the problem in this research such **methods** were used: analysis and synthesis, system analysis, comparison, logical generalization of results, design of logical data structures.

Scientific novelty. The scientific novelty of the thesis is that the system developed is the first emotional neural dialogue system for recommending online courses. In it, outside from the communication on educational topics, you can control the emotional responses of the system, which was not possible in previous developments related to neural dialogue systems. Potential applications and practical value of the results of the thesis:

1) The developed mechanism of emotionality can be applied to improve existing neural dialogue systems and applications based on them;

2) The developed method for controlling the emotional coloring of answers can

be extended to control other aspects of natural communication, for example, style of speech;

3) This system can be introduced as an additional resource of searching for materials in educational systems.

Approbation of the results. The results of the study were presented at the 2017 by the IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON).

Publications:

1. Shaptala R. Exploring the vector space model for online courses / R. Shaptala, A. Kyselova, G. Kyselov. // 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON). – 2017. – С. 861–864.
2. Прийнято: Shaptala R. Evaluation of approaches to emotion classification / R. Shaptala, A. Kyselova, G. Kyselov. // System analysis and information technology: 20-th International conference SAIT 2018, Kyiv, Ukraine.
3. Подано: Shaptala R. Neural dialogue system with emotion embeddings / R. Shaptala, A. Kyselova, G. Kyselov. // IEEE First International Conference on System Analysis & Intelligent Computing (SAIC). – 2018.

Keywords: neural dialogue system, emotionality, recurrent neural network, FastText, entity vector.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ВЛ	Випадкові ліси
ДР	Дерева рішень
ДС	Діалогова система
ЕНДС	Емоційна нейронна діалогова система
ЛР	Логістична регресія
МГК	Метод головних компонентів
НДС	Нейронна діалогова система
РНМ	Рекурентна нейронна мережа
FT - FastText	Безкоштовна, легка бібліотека, яка дозволяє користувачам навчати вектори слів та класифікатори тексту.
t-SNE - t-distributed stochastic neighbor embedding	t-розподілене стохастичне сусіднє представлення

ВСТУП

Діалогова система (ДС) - це комп'ютерна система, призначена для спілкування з людиною. Така система намагається імітувати людське спілкування для ефективної взаємодії з людиною. Нейронна ДС - це діалогова система побудована за допомогою технологій машинного навчання, а саме - нейронних мереж.

Актуальність теми дослідження полягає в тому, що поточні діалогові системи охоплюють лише семантичну та граматичну складові природньої мови, ігноруючи емоційне забарвлення необхідне при живому спілкуванні, а тому вимагають механізмів інтеграції емоційності при розмові. Крім того, здатність надати таким системам специфічних знань та можливості здійснювати обдуманий пошук та рекомендації є дуже важким завданням та потребує подальших досліджень. Таким чином створення діалогової системи, яка може імпонувати емоціям користувача та має знання про онлайн курси, здатне сприяти прискореному навчанню фахівців у певній області.

Метою дипломної роботи є створення емоційної нейронної діалогової системи зі знаннями для пошуку онлайн курсів. Поставлена мета вимагає вирішення наступних наукових задач:

1) аналіз та класифікація емоцій у текстових даних, зокрема діалогових репліках;

2) розробка механізму емоційності у нейронних діалогових системах;

3) розробка семантичної моделі пошуку онлайн курсів за їх текстовим описом.

Об’єктом дослідження є процес текстової взаємодії людина-машина.

Предметом дослідження є діалогова система з емоціями та знаннями на тему онлайн курсів і здатністю рекомендувати їх.

Досягнення поставленої мети реалізовано з використанням методів інтелектуального аналізу даних, обробки природньої мови та машинного навчання, в тому числі глибинного навчання - рекурентних нейронних мереж та векторів сутностей.

Наукова новизна дипломної роботи полягає в тому, що розроблена система є першою емоційною нейронною діалоговою системою для рекомендації онлайн курсів. У ній при спілкуванні на сторонні від освіти теми, можна контролювати емоційність відповідей даної системи, що не було можливим у попередніх розробках, пов’язаних з НДС.

Потенційні застосування та практична цінність результатів дипломної роботи:

1) Розроблений механізм емоційності можна застосувати для покращення існуючих нейронних діалогових систем та додатків на їх основі;
2) Розроблений метод контролю емоційного забарвлення відповідей можна розширити і для контролю інших аспектів природнього спілкування, наприклад стилю мовлення;
3) Дану систему можна запровадити як додатковий ресурс пошуку матеріалів у освітніх системах.

1 ЕМОЦІЙНА НЕЙРОННА ДІАЛОГОВА СИСТЕМА ЗІ ЗНАННЯМИ

1.1 ОПИС НЕЙРОННИХ ДІАЛОГОВИХ СИСТЕМ

Діалогова система - це система, яка інтерпретує та реагує на заяви, зроблені користувачами звичайною природною мовою. Вона інтегрує методи обчислювальної лінгвістики з спілкуванням через Інтернет. Система природного діалогу - це форма діалогової системи, яка намагається покращити легкість використання та задоволеність користувачів, імітуючи поведінку людей.

Такі системи бувають двох типів:

1. Моделі, що базуються на інформаційному пошуці - використовують сховище попередньо визначених відповідей та певні евристичні методи для вибору відповіді на основі контексту. Евристичний метод може бути досить простим і базуватись, як правило, на основі співпадіння слів та виразів, або настільки ж складним, як ансамбль класифікаторів машинного навчання. Ці системи не генерують нового тексту, вони просто вибирають відповідь із фіксованого набору[1].
2. Генеративні моделі - не покладаються на заздалегідь визначені відповіді. Вони генерують нові відповіді з нуля. Генеративні моделі, як правило, засновані на техніці машинного перекладу, але замість перекладу з однієї мови на іншу, ми "перекладаємо" від початкової репліки до наступної (відповіді)[1].

Нейронні діалогові системи - текстові діалогові системи, призначені для взаємодії людина-машина, і є підвидом генеративних моделей. Різниця між НДС та іншими видами генеративних ДС полягає в тому, що перші побудовані виключно на основі такого класу алгоритмів як нейронні мережі. Місце НДС в ієрархії ДС представлене на рисунку 1.1.

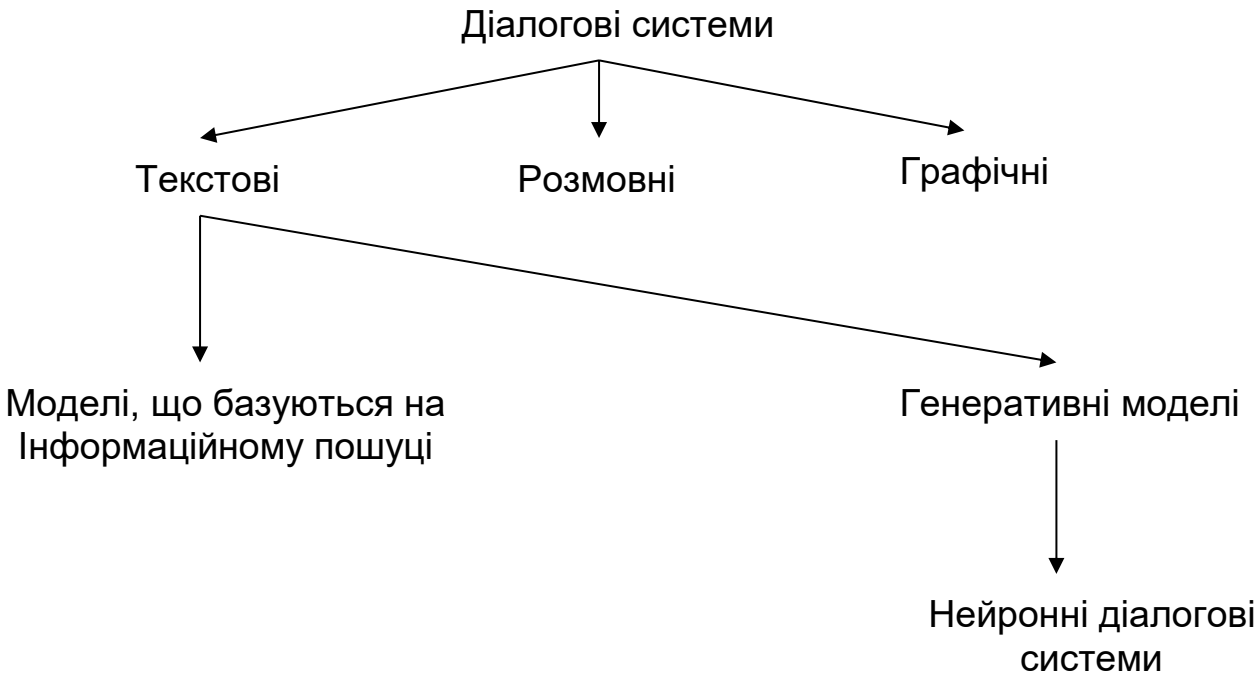


Рисунок 1.1 – НДС в ієрархії діалогових систем

1.2 ЕМОЦІЙНІСТЬ В НЕЙРОННИХ ДІАЛОГОВИХ СИСТЕМАХ

Як життєво важлива частина людського інтелекту, емоційний інтелект визначається як група ментальних здібностей, які беруть участь в усвідомленні та розумінні власних емоцій і емоцій оточуючих[2]. Люди з високим рівнем емоційного інтелекту добре розуміють свої емоції і почуття інших людей, можуть ефективно керувати своєю емоційною сферою, і тому в суспільстві їхня поведінка більш адаптивна і вони легше досягають своїх цілей у взаємодії з оточуючими. Довгострокова мета штучного інтелекту - дати можливість машинам розуміти емоції. Саме тому створення діалогового агента, що здатний спілкуватися з користувачем на рівні людини, необхідно обладнати машину можливістю сприймання та вираження емоцій.

Наскільки нам відомо, фактор емоцій не розглядався в існуючих нейронних моделях для генерації бесід. Машини потребують моделі емоцій для синтезу емоцій і їх висловлення. Модель емоцій повинна дозволити використовувати емоції у діалозі, як це роблять люди. Подія, яка засмучує людей, наприклад, втрата грошей, також

повинна засмутити і машину. Модель емоцій повинна вміти оцінювати всі ситуації, в яких може перебувати співрозмовник. Неможливість розмовного агента виявити емоційний стан співрозмовника може бути інтерпретована ним як відсутність симпатії[3].

1.3 ЗНАННЯ У ДІАЛОГОВИХ СИСТЕМАХ

Сучасні діалогові системи можуть бути повністю керовані даними, без ручного кодування. Однак ці повномасштабні системи не мають фундаменту в реальному світі і доступу до будь-яких зовнішніх знань (текстових чи структурованих). Попри те, що відповіді, які виробляють нейронні моделі розмовно підходять, вони рідко включають фактичний вміст. Це контрастує з природнім поняттям діалогу, у який в будь-який момент можна легко ввести об'єкти та факти. Саме тому, на даний момент, переважають традиційні діалогові системи, засновані на правилах, за умови, що від системи вимагаються конкретні знання чи факти. Недоліком такого підходу є те, що часто за рахунок значного ручного кодування, такі системи важко масштабувати на нові прикладні області чи завдання.

Іншим підходом для надання знань нейронній діалоговій системі є ідентифікація і підтримка фактів у пам’яті системи під час навчання її мові. Такий метод інкорпорації знань маловивчений і потребує великої кількості специфічних даних.

1.4 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Нейронні діалогові системи часто є предметами публікацій на наукових конференціях з машинного навчання в останні роки. Стаття [4] - одна з перших в області нейронних діалогових систем і показує як створити найпростішого чат-бота на основі даних за допомогою методу seq2seq. Усі наступні статті спираються на даний підхід, адже він є фундаментальним у побудові таких систем. Запропонована архітектура нейромережі та спосіб генерації відповідей дозволяє розробляти діалогові системи без застосування вручну прописаних правил, а навчати програму мові автоматично - з даних.

Поки що єдиною публікацією на тему емоційних діалогових систем є [5], у якій автори показують можливість створення такої системи. Проблеми з даним підходом: система не прив’язана до конкретного завдання і не має специфічних знань; дані, що використовувались для тренування - китайською та не відкриті. Автори запевняють, що їх система може генерувати відповіді у таких стилях: like(подобається), happy(щаслива), sad(сумна), anger(зла), disgust(відразлива).

У статті [6] автори створили діалогову систему, яка спирається на базу знань про ресторани і при спілкуванні може запропонувати спробувати певну страву з певного закладу. Проблема: відповіді системи виходять надто короткі та мало імпонують користувачам.

Схожою проблемою до побудови діалогових систем є машинний переклад. Саме розвиток даної області призвів до більшості відкриттів у обробці натуральної мови. Стаття [7] показує кілька важливих підходів для побудови масштабованої системи машинного перекладу, які можна перенести на область проектування діалогових систем. А саме: багатошаровий seq2seq, Attention та квантизація відповідей для прискорення практичної роботи алгоритму.

Проблема, яка наявна у всіх нейронних діалогових системах, - це короткі та не інформативні відповіді, так як при навчанні вони були найбільш прийнятними. Без контексту, дані відповіді гарно підходять під більшість запитань, але з впровадженням контексту стає видно, що модель віддає загальні фрази. Щоб побороти дане явище автори [8] запропонували використовувати функцію тренування, яка б заохочувала різноманіття у відповідях. Недоліком даного підходу є важка імплементація та ресурси, потрібні для цього. Для вирішення даної проблеми автори [9] запропонували підходи glimpse model та stochastic beam-search, які дозволяють ефективно генерувати довгі та релевантні відповіді на питання.

Стаття [10] намагається подолати іншу проблему чат-ботів - відсутність власної ідентифікації. Наприклад, якщо діалоговій системі двічі поставити запитання в якому місті вона живе, відповідь буде різна. Автори показують, що можливо створити таку

систему, яка буде консистентно відповідати на такі питання, “позичаючи” ідентичність осіб, які були в її тренувальному наборі.

1.5 ВИСНОВОК

Діалогові системи мають велике різноманіття та довгу історію. Сучасні дослідження в даній сфері мало фокусуються на емоційному інтелекті нейронних діалогових систем, тому дана робота націлена на покращення даної ситуації. Крім того, дуже важливо, щоб діалоговий агент мав прикладне значення, а тому неодмінно повинен мати знання. Для експериментів з впровадження знань у ЕДС було обрано область пошуку онлайн курсів через доступність даних про них та велику практичну цінність. Отже, важливо запропонувати модель, яка зможе генерувати відповіді, що підходять не тільки за змістом, але і з заданим емоційним забарвленням.

Створення емоційної нейронної діалогової системи зі знаннями для пошуку онлайн курсів є актуальним завданням, яке можна вирішити з’єднавши різні традиційні та новітні методи та підходи. Виходячи з аналізу літератури та попередніх робіт, було вирішено об’єднати сучасний підхід рекурентних нейронних мереж для створення емоційної діалогової системи з традиційним підходом, заснованим на правилах, для насичення системи фактами про онлайн курси.

2 РОЗРОБКА ЕМОЦІЙНОЇ НЕЙРОННОЇ ДІАЛОГОВОЇ СИСТЕМИ ЗІ ЗНАННЯМИ

2.1 РОЗРОБКА КЛАСИФІКАТОРА ЕМОЦІЙ

2.1.1 ПОТРЕБА В КЛАСИФІКАТОРІ ЕМОЦІЙ

Високоякісні дані, анотовані емоціями, важко отримати у великому обсязі, адже анотація емоцій - це досить суб'єктивне завдання і через свою складність вимагає багато часу та людських ресурсів. Крім того, на даний момент не існує відкритих діалогових даних розмічених емоціями.

Для отримання великої кількості даних з емоціями для ЕНДС, ми навчаємо класифікатор емоцій на основі вручну анотованих корпусів, які знаходяться у вільному доступі. Класифікатор в подальшому використовується для автоматичної анотації діалогів для навчання ЕНДС.

Недоліком такого підходу є наявність помилок у класифікації, а отже велика зашумленість вихідних даних, які підуть на вхід діалоговій системі. Але враховуючи дороговизну власної розмітки діалогів емоціями, штучний автоматичний класифікатор - єдина оптимальна опція.

2.1.2 НАБОРИ ДАНИХ ДЛЯ ТРЕНУВАННЯ

Для побудови класифікатора емоцій у текстових даних було використано 6 різних джерел текстів, анотованих емоціями. Кожен з наборів даних описаний нижче:

- Crowdflower's The Emotion in Text[11] - набір даних, що задумувався як варіанті популярної задачі аналізу настроїв; містить анотації емоційного змісту (наприклад, щастя, смуток і гнів) текстів. Сотні тисяч прикладів 13 класів. Підмножина цих даних використовується в експерименті, який Crowdflower завантажили в Microsoft Cortana Intelligence Gallery;
- Hashtag Emotion Corpus[12] (також відомий як Twitter Emotion Corpus, TEC) - набір даних, що складається з твітів з емоційними хештегами. Використовувався при створенні NRC Hashtag Emotion Lexicon[13];

- Колекція любовних листів, ненависних листів та передсмертних записок[14];
- WASSA-2017 Shared Task on Emotion Intensity (EmoInt)[15] - частина восьмого семінару Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA-2017). Навчальні та тестові набори надаються для чотирьох емоцій: радості, смутку, страху та гніву. Наприклад, набір даних, анотований як гнів, містить твіти разом з оцінкою від 0 до 1, що вказує на ступінь гніву, який відчуває оратор. Дані тесту включають лише текст твіта. Твіти, оцінка яких була менша 0.5 не включались для тренуваня класифікатора;
- Movie reviews, annotated for emotion classification[16] - дані складаються з аматорських рецензій фільмів, отриманих від IMDB та анотованих для проекту "Пошук публічних дискурсів" Амстердамського університету та Центру електронних наук Нідерландів;
- Affective Text: Data Annotated for Emotions and Polarity[17] - набір даних, що складається з 1000 тестових заголовків та 200 заголовків розробки, кожен з яких анотований з шістьма емоціями та орієнтацією полярності.

Кількість екземплярів з кожного набору даних, а також відносний відсоток вмісту кожного з наборів даних у фінальному показані на рисунку 2.1.

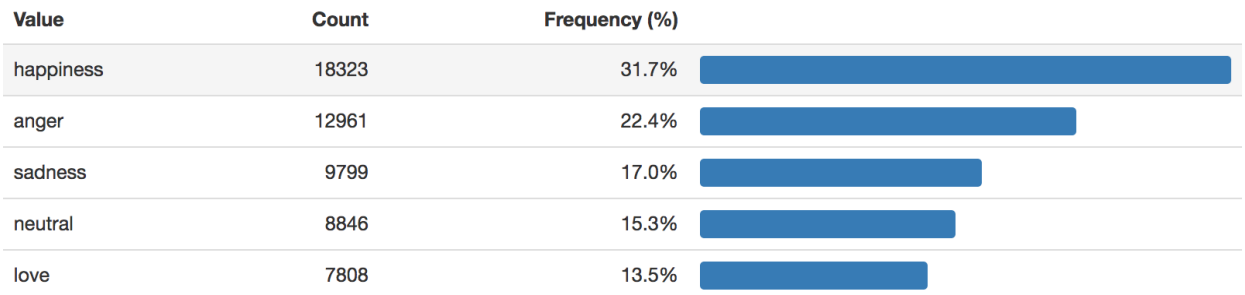
Value	Count	Frequency (%)
crowdflower	36816	63.8%
hashtag_emotion	14254	24.7%
love_letters	3819	6.6%
wassa	2207	3.8%
spudisc	350	0.6%
affective_text	291	0.5%

Рисунок 2.1 - Джерела даних для тренування класифікатора

Для зведення усіх даних до спільного набору класів були проведені:

1. Перейменування класів - частина наборів даних мають різні позначення для одної і тої ж емоції, наприклад: sad - sadness, happy - happiness;
2. Об’єднання класів worry, hate з anger;
3. Відфільтровування класів, як частка яких у тренувальному наборі менша трьох відсотків;
4. Ручний перегляд окремих випадків повтору екземплярів з різними анотаціями.

Фінальний набір даних для побудови класифікатора емоцій мав розподіл класів зображений на рисунку 2.2.



Value	Count	Frequency (%)
happiness	18323	31.7%
anger	12961	22.4%
sadness	9799	17.0%
neutral	8846	15.3%
love	7808	13.5%

Рисунок 2.2 - Розподіл класів у наборі даних для класифікатора емоцій

2.1.3 АЛГОРИТМИ КЛАСИФІКАЦІЇ

2.1.3.1 ДЕРЕВА РІШЕНЬ

Дерева рішень (інші назви: дерева класифікацій, регресійні дерева) — тип моделей, що використовується в галузі статистики та аналізу даних для прогнознозування та класифікації. Дерево складається з набору листів та гілок. На ребрах («гілках») дерева рішень знаходяться атрибути, від яких залежить функція втрат, в «листах» знаходяться значення даної функції, а в інших вузлах — атрибути, за якими відрізняються екземпляри. Щоб класифікувати новий екземпляр, потрібно спуститися по дереву відповідно до його характеристик до листа і видати відповідне значення. Дерева рішень широко використовуються в інтелектуальному аналізі даних, так як мають просту інтерпретацію, в той же час досягаючи високої точності на задачах машинного навчання[18].

Кожен лист являє собою значення цільової змінної, тобто класом, яке залежиться від попередніх вузлів при русі від кореня. Дерево може бути навченим

через поділ вихідних наборів змінних на підмножини, що базуються на зміні значень вхідних змінних. Такий процес буде повторюватись всередині кожної з отриманих підмножин. Такі рекурсивні дії закінчуються тоді, коли підмножина має ті ж значення цільової змінної, а отже, не додає цінності для прогнозу чи класифікації. Процедура «згори донизу» є прикладом жадібного алгоритму, і на сьогоднішній день є найбільш поширеною стратегією побудови дерев рішень, але є і інші[18]. В машинному навчанні, дерева рішень можуть використовуватись в якості математичних та програмних методів, для допомоги в описі, класифікації і узагальненні набору даних.

Загальний алгоритм побудови дерева рішень за тестовими прикладами виглядає таким чином:

- Вибираємо черговий атрибут x, поміщаємо його в корінь.
- Для всіх його значень i:
  - Залишаємо з тестових прикладів тільки ті, у яких значення атрибута x дорівнює i
  - Рекурсивно будуємо дерево в цьому нащадку

Вибір чергового атрибута зазвичай здійснюється на підставі коефіцієнту Джині(9) або на підставі ентропії чи приросту інформації(10).

$$G = 1 - \sum_{j=1}^{c} p_j^2, \quad (9)$$

де c – кількість класів;

p_j – частка записів з класу j у наборі даних.

$$H = -\sum_{j=1}^{c} p_j \log(p_j). \quad (10)$$

До інших способів вибору атрибутів відносяться:

- Алгоритм C4.5, де вибір атрибута відбувається на підставі нормалізованого приросту інформації.
- Алгоритм CART і його модифікації.
- Автоматичний детектор взаємодії Хі-квадрат (CHAID). Виконує багаторівневий поділ при розрахунку класифікації дерев[18].

Практично, використання останніх алгоритмів часто призводить до явища перенавчання, коли алгоритм запам’ятовує вхідний набір даних і не узагальнюється до нових екземплярів.

Серед інших методів інтелектуального аналізу даних, дерева рішень мають такі переваги:

- Прості для розуміння і інтерпретації. Люди можуть зрозуміти моделі дерева рішень після короткого пояснення;
- Вимагають невеликої підготовки даних. Інші методи часто вимагають нормалізації даних, введення фіктивних змінних і видалення порожніх значень;
- Можливість обробляти як числові так і категоріальні дані. Інші методи, як правило, спеціалізуються на аналізі масивів даних, які мають тільки один тип змінної (наприклад, правила співвідношення можуть бути використані тільки з номінальним типом змінних в той час як нейронні мережі можуть бути використані тільки з числовими змінними);
- Використовує модель білого ящика. Якщо дана ситуація спостерігається в моделі, пояснення цього легко випливає через булеву логіку. На противагу цьому, в моделі чорного ящика, пояснення результатів, як правило, важко зрозуміти, як наприклад, за допомогою штучної нейронної мережі.
- Можливість перевірки моделі з використанням статистичних тестів. Це дозволяє обгрунтувати надійність моделі.
- Стійкі. Працюють добре, навіть якщо припущення, на яких вони базуються, дещо порушуються, на відміну від істинної моделі, з якої були отримані дані.
- Добре працює з великими наборами даних. Великі обсяги даних можуть бути проаналізовані з використанням стандартних обчислювальних ресурсів в розумний часовий термін.

2.1.3.2 ВИПАДКОВІ ЛІСИ

Класифікатор випадкових лісів – ансамблевий алгоритм. Ансамблеві алгоритми - це ті, що поєднують в собі більше, ніж один алгоритм одного й іншого виду для класифікації об'єктів. Наприклад, після обчислання прогнозу алгоритмами Наївного

Байєса, SVM та Деревом Рішень проводиться голосування для прийняття остаточного рішення про клас для тестового об'єкта.

Випадковий ліс - це гнучкий, простий у використанні алгоритм машинного навчання, який навіть без налаштування гіперпараметрів дає чудовий результат більшу частину часу. Це також один з найпоширеніших алгоритмів через його простоту і той факт, що він може бути використаний як для класифікації, так і для регресії.

У ході побудови випадкового лісу створюється набір дерев рішень з довільно вибраних підмножин навчального набору для покращення точності прогнозування та контролю перенавчання. Розмір підмножин завжди збігається з розміром вибірки вхідних даних, але підмножини обираються з заміною. Потім проводиться об'єднання голосів різних дерев рішень, щоб визначити остаточний клас тестового об'єкта.

Крім того, при поділі вузла під час побудови дерева, вибраний розділ стає більше не найкращим поділом між усіма ознаками. Замість цього вибраний розділ - це найкращий розподіл між випадковою підмножиною ознак. Внаслідок цієї випадковості зміщення випадкового лісу зазвичай трохи збільшується (з урахуванням упередженості одного невипадкового дерева), але за рахунок усереднення його дисперсія також зменшується, що як правило, більше, ніж компенсує зміщення, а отже, дає загально кращу модель.

Основними параметрами, що коригуються при використанні цих методів, є кількість дерев і максимальна кількість ознак. Перший - кількість дерев у лісі. Чим більше, тим краще, але і довше буде потрібно обчислювати прогноз. Крім того, слід зазначити, що після того, як буде пройдена критична кількість дерев, результати перестануть суттєво покращуватись. Останній - це розмір випадкових підмножин ознак, які слід враховувати при поділі вузла. Чим нижче, тим більше зменшення дисперсії, але також більший приріст зміщення. Емпіричні хороші значення за замовчуванням - це максимальна кількість ознак рівна повній кількості ознак для регресійних завдань, або корінь з повної кількості ознак для завдань класифікації. Гарні результати часто досягаються при не обмеженні максимальної глибини дерев у

поєднанні з мінімальною кількістю екземплярів для поділу рівній двом (тобто, коли повністю дерева будуються повністю). Однак варто враховувати, що ці значення, як правило, не оптимальні, і можуть призвести до моделей, що споживають багато оперативної пам'яті. Найкращі значення параметрів завжди повинні бути отримані за допомогою перехресної перевірки.

За винятком деяких випадків класифікатор випадкових лісів має всі гіперпараметри класифікатора дерева рішень, а також всі гіперпараметри класифікатора, для управління самим ансамблем. Дерева рішень можуть постраждати від перенавчання. Випадковий ліс зазвичай запобігає перенавчанню, створюючи випадкові підмножини функцій і створюючи менші дерева за допомогою цих підмножин. Це не працює кожного разу, і також робить обчислення повільнішим, залежно від того, скільки дерев створює випадковий ліс.

Оптимальне число дерев підбирається таким чином, щоб мінімізувати помилку класифікатора на тестовій вибірці. Випадкові ліси, отримані в результаті застосування технік, описаних раніше, можуть бути природним чином використані для оцінки важливості змінних в задачах регресії та класифікації. Перший крок в оцінці важливості змінної в тренувальному наборі — тренування випадкового лісу на цьому наборі. Під час процесу побудови моделі для кожного елемента тренувального набору вважається так звана out-of-bag — помилка. Потім для кожної сутності така помилка опосередковується по всьому випадковому лісі[19].

Відносний рейтинг (тобто глибина) ознаки, яка використовується як вузол рішення в дереві, може використовуватися для оцінки відносної важливості цієї ознаки щодо передбачуваності цільової змінної. Функції, що використовуються у верхній частині дерева, сприяють остаточному рішенню прогнозу більшої частки вхідних зразків. Очікувана частка зразків, які вони внесли, може бути використана як оцінка відносної важливості ознак.

2.1.3.3 ЛОГІСТИЧНА РЕГРЕСІЯ

У статистиці багатокласова логістична регресія є класифікаційним методом, який узагальнює логістичну регресію на багатокласові проблеми, тобто з більш ніж

двома можливими дискретними анотаціями[20]. Тобто це модель, яка використовується для прогнозування ймовірностей різних можливих результатів поліноміально розподіленої залежної змінної, на основі незалежних змінних (які можуть бути дійсними, бінарними чи категоріальними).

Ідея методу, як і в багатьох інших статистичних методах класифікації, полягає в тому, щоб побудувати лінійну функцію предиктора, який буде видавати оцінку на основі набору ваг, які лінійно поєднуються з незалежними змінними даного спостереження за допомогою скалярного добутку(2.1).

$$Оцінка(X_i, k) = \beta_k * X_i, \quad (2.1)$$

де Xi - вектор незалежних змінних, що описують спостереження і, βk - вектор ваг (або коефіцієнтів регресії), що відповідає класу k, а Оцінка(Xi, k) - оцінка, пов'язана з призначенням спостереження і до категорії k. Прогнозований результат - це результат з найвищою оцінкою.

Різниця між багатокласовою логіт-моделлю та іншими методами, моделями, алгоритмами класифікації (методом опорних векторів, дискримінантним аналізом і т. д.) полягає в процедурі визначення (тренування) оптимальних ваг / коефіцієнтів та методі інтерпретації оцінок. Зокрема, у багатокласовій логістичній регресії оцінка може бути безпосередньо перетворена в значення ймовірності, що вказує на вірогідність спостереження і відповідати класу k за даними незалежних характеристик спостереження. Це забезпечує принциповий спосіб включення прогнозу конкретної моделі в більшу процедуру, яка може включати кілька таких прогнозів, кожен з можливістю помилок. Без таких засобів поєднання прогнозів помилки, як правило, помножуються.

2.1.3.4 FASTTEXT

FastText - це безкоштовна, легка бібліотека, яка дозволяє користувачам навчати вектори слів та класифікатори тексту. Зі зростанням кількості онлайн-даних існує потреба у більш гнучких інструментах для кращого розуміння вмісту дуже великих наборів даних, щоб забезпечити більш точні результати класифікації. Для вирішення цієї проблеми, лабораторія Facebook AI Research (FAIR) - відкрила доступ до FastText,

бібліотеки, призначеної для створення масштабованих рішень для векторизації та класифікації тексту.

FastText поєднує в собі деякі з найуспішніших концепцій, які впродовж останніх декількох десятиліть впроваджуються спільнотою обробки природної мови та машинного навчання. До них відносяться вектори з сумою слів і набори n-грамів, а також використанням підслів та обміну інформацією між класами за допомогою прихованих векторів. Також у бібліотеці наявне використання ієрархічного софтмаксу при незбалансованому розподілі класів для прискорення обчислень. Всі ці різні підходи використовуються для двох різних завдань: ефективної класифікації тексту та вивчення векторних представлень слів.

Глибокі нейронні мережі останнім часом стали дуже популярними для обробки тексту. Хоча ці моделі досягають дуже високої продуктивності в умовах обмеженої лабораторної практики, вони можуть бути повільними для навчання та тестування, що обмежує їх використання на дуже великих наборах даних.

FastText допомагає вирішити цю проблему. Щоб бути ефективним на наборах даних з дуже великою кількістю категорій, він використовує ієрархічний класифікатор, а не плоску структуру, в якій різні категорії організовані в дереві (бінарне дерево замість списку). Це зменшує часові складності підготовки та тестування класифікаторів тексту з лінійної до логарифмічної з урахуванням кількості класів. FastText також використовує той факт, що класи незбалансовані (деякі класи з'являються частіше, ніж інші), використовуючи алгоритм Хаффмана для побудови дерева, що використовується для представлення категорій. Тому глибина дерева дуже частих категорій є меншою, ніж для нечастих, що призводить до подальшої обчислювальної ефективності.

FastText також представляє текст за допомогою маломірного вектора, який отримується підсумовуванням векторів, відповідних словам, що містяться в тексті. У бібліотеці з кожним словом лексики пов'язаний невеликий за розміром вектор. Це приховане представлення розподіляється між усіма класифікаторами для різних категорій, що дозволяє отримувати інформацію про сприйняті слова для однієї

категорії іншими категоріями. Такі види представлень, названі bag of words, ігнорують порядок слів. У FastText також використовуються вектори для наборів послідовних слів - ngrams, щоб враховувати місцевий порядок слів, що важливо для багатьох проблем текстової класифікації[21].

2.1.4 РЕЗУЛЬТАТИ

Точність (precision) і повнота (recall) є метриками які використовуються при оцінці більшості алгоритмів класифікації. Іноді вони використовуються самі по собі, іноді в якості базису для похідних метрик, таких як F1-score. Суть точності і повноти дуже проста. Точність системи в межах класу - це частка документів, які дійсно належать даному класу, щодо всіх документів які система віднесла до цього класу. Повнота системи - це частка знайдених класифікатором документів, які належать класу, щодо всіх документів цього класу в тестовій вибірці.

Зрозуміло що чим вищі точність і повнота, тим краще. Але в реальному житті максимальна точність і повнота недосяжні одночасно і доводиться шукати якийсь баланс. Тому, хотілося б мати якусь метрику яка об'єднувала б у собі інформацію про точність та повноту нашого алгоритму. Саме такою метрикою є F1-score. F1-score - це гармонійне середнє між точністю і повнотою. Вона прагне до нуля, якщо точність або повнота прагне до нуля. Саме тому ми обираємо дану метрику для оцінки роботи класифікатора.

Наведемо деталі тренованих алгоритмів:

- Логістичній регресії на вхід передавались TF-IDF (term frequency-inverse document frequency) ознаки, виділені з текстів; для врахування багатьох класів логістична регресія була натренована методом OVR(one-vs-rest), що означає, що було натреновано кілька бінарних моделей, кожна з яких оцінює власну категорію; для боротьби з перенавчанням було використано L2 регуляризацію з оберненим коефіцієнтом сили - 0.9.
- Дерево рішень мало такі гіперпараметри: максимальна глибина - 10, критерій - ентропія, мінімальна кількість екземплярів для розділу внутрішнього вузла - 3.

- Випадкові ліси мали такі гіперпараметри: кількість дерев - 10, мінімальна кількість екземплярів для розділу внутрішнього вузла - 2, критерій - Джині.
- FastText тренувався з наступними гіперпараметрами: розмірність векторів - 500, темп навчання - 0.01, кількість епох - 50, кількість негативних екземплярів - 100.

Всі гіперпараметри були обрані за допомогою сіткового пошуку з 10-кратною перехресною валідацією. Фінальні результати наведено у таблиці 2.1. Як бачимо, найкращу оцінку здобув метод класифікації з бібліотеки FastText, тому обираємо його для анотації діалогових даних.

Таблиця 2.1 – F1-micro оцінка алгоритмів класифікації емоцій

Алгоритми	**F1-micro**
Дерева Рішень	0.3755
Випадкові Ліси	0.4063
Логістична Регресія	0.5028
FastText	0.5118

2.2 РОЗРОБКА ЕМОЦІЙНОЇ ДІАЛОГОВОЇ СИСТЕМИ

2.2.1 SEQ2SEQ МОДЕЛІ

Seq2Seq моделі - найбільш часто використовувана архітектура в машинному перекладі і нейромережевих системах типу питання-відповідь. Найбільша кількість пам'яті в таких моделях витрачається на зберігання матриці представлень, що містить репрезентації кожного слова зі словника. Для послівної генерації узгодженої відповіді потрібно мати поряд зі стандартною формою слова ще й всі його словоформи. В деяких мовах у слів є лише невелика кількість словоформ (наприклад, однина і множина), що зменшує кількість пам’яті, потрібної для зберігання векторів слів.

Моделі зі словниками, які повністю покривають всі словоформи, перевищують розумні обмеження як за часом, так і по пам'яті. Для обходу цієї проблеми часто використовують посимвольні моделі. У таких моделях розмір словника відповідає розміру використовуваного алфавіту. Посимвольна генерація дозволяє уникати зберігання словоформ, однак через збільшення довжини послідовності в кілька разів, модель швидко забуває початок тексту. Також ці моделі, хоч і здатні генерувати граматично правильні слова, не можуть створювати семантично правильні речення. Саме тому посимвольна модель не підходить для створення емоційної нейронної діалогової системи.

У даній роботі була створена звичайна Seq2seq архітектура для того, щоб мати відправну точку та засіб якісної та кількісної оцінки емоційної системи. Така модель складається з двох рекурентних мереж: кодер і декодер. Кодер будує представлення вхідної послідовності слів. Далі отримане представлення передаються в декодер. З його допомогою декодер намагається відновити цільову послідовність слів. У завданнях машинного перекладу вхідною і вихідною послідовностями є текст та його переклад на цільову/цільові мови. У діалогових системах - це питання і відповідь.

Для перетворення слів у вхідні вектора використовується так звана матриця представлень (embedding matrix). Кількість рядків цієї матриці дорівнює розміру словника, а число стовпців - задається користувачем при побудові моделі і є одним із гіперпараметрів. Кожне слово перед подачею на вхід комірці пам’яті замінюється на відповідний рядок матриці представлень.

Серед комірок пам’яті однією з найбільш ефективних і широковживаних є LSTM. Long-short term memory (LSTM) [22] - популярна архітектура рекурентної мережі. Основна ідея цієї архітектури - виділення комірки пам'яті, відповідальної за збереження інформації, отриманої в попередні моменти часу. Для керування даною коміркою виділяються три шлюзи: вхідний і (контролює надходження даних в пам'ять), що вихідний o (контролює вихід даних з пам'яті) і шлюз пам'яті f (відповідальний за збереження / забування попереднього стану комірки).

LSTM мережі користуються великою популярністю, так як вони здатні виявляти довгострокові залежності в даних. Наступні формули описують процес обчислення однієї ітерації використаної в роботі архітектури LSTM мережі:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t);$$

де $\mathbf{x}_t \in \mathbb{R}^d$ вхідний вектор, $\mathbf{f}_t \in \mathbb{R}^h$ активаційний вектор шлюзу пам’яті активаційний вектор вхідного шлюзу, $\mathbf{o}_t \in \mathbb{R}^h$ активаційний вектор вихідного шлюзу, $\mathbf{h}_t \in \mathbb{R}^h$ вихідний вектор комірки, $\mathbf{c}_t \in \mathbb{R}^h$ вектор стану комірки, $\mathbf{W} \in \mathbb{R}^{h \times d}$,$\mathbf{U} \in \mathbb{R}^{h \times h}$ and $\mathbf{b} \in \mathbb{R}^h$- ваги, що будуть натреновані у процесі навчання, t - поточний часовий крок, ○ – добуток Адамара.

Формули 1-5 відповідають схематичному рисунку 2.3.

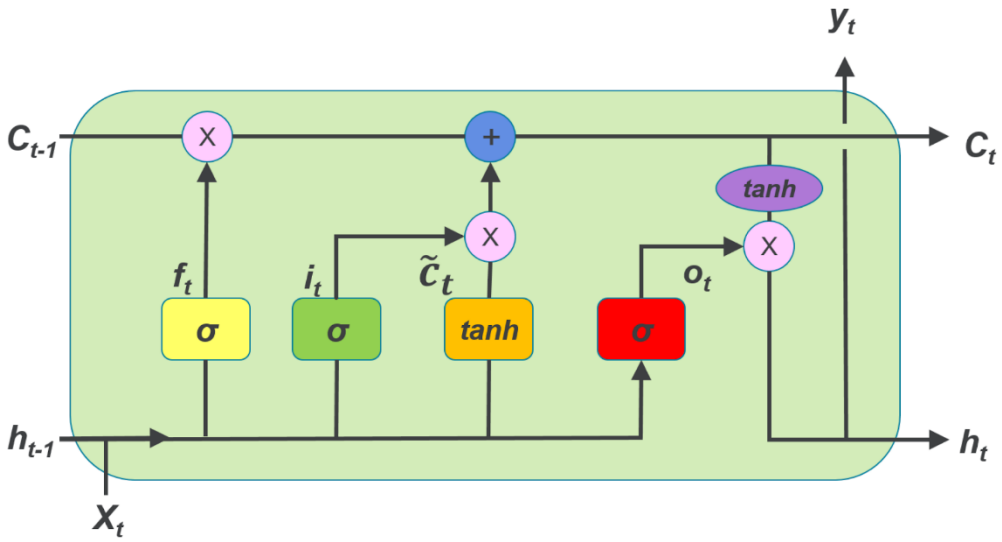


Рисунок 2.3 - Архітектура LSTM комірки

LSTM мережі навчаються за допомогою алгоритму зворотного поширення помилки крізь час (backpropagation through time), ідея якого полягає в розгортанні графа обчислень в часі. З LSTM комірок можна вибудовувати багатошарові нейронні мережі, передаючи вихідну послідовність чергового шару на вхід наступного.

На вхід декодеру на першому такті подається спеціальний символ <SOS>(start of sequence - початок послідовності), потім на кожному такті подається згенроване на попередній ітерації слово. Генерація відповіді триває до тих пір, поки не буде згенеровано спеціальне слово - маркер кінця рядка <EOS> (end of sequence - кінець послідовності). Розподіл передбачених слів передається в функцію втрат. Під час передбачення потрібно знайти найбільш ймовірне слово. Зробити це безпосередньо неможливо, так як модель дозволяє обчислювати тільки найкраще слово при фіксованих попередніх. Компромісом між жадібним вибором слів і повним перебором є променевий пошук - Beam Search. При використанні цього методу на кожній ітерації вибирається невелика кількість кращих кандидатів, а решта гіпотез відкидаються. Променевий пошук використовує пошук в ширину, щоб побудувати своє дерево пошуку. На кожному рівні дерева він генерує всіх нащадків станів на поточному рівні, сортуючи їх в порядку збільшення евристичної ваги (значимості). Тим не менш він зберігає тільки певну кількість станів на кожному рівні (так звана ширина променя). Чим більша у пучка променів ширина, тим менше станів видаляється. З нескінченою шириною променя жоден стан не видаляється і

променевий пошук ідентичний пошуку в ширину. Ширина пучка обмежує об’єм пам’яті, необхідний для виконання пошуку. Оскільки цільовий стан потенційно може бути видалений, променевий пошук жертвує повнотою (гарантія того, що алгоритм буде припинений з розв’язком, якщо він існує) та оптимальністю (гарантія того, що він знайде найкращий розв’язок)[23].

Вчительський примус(Teacher forcing)[24] - це метод швидкого та ефективного навчання рекурентних нейронних мереж, які використовують вихід з попереднього етапу роботи в якості вхідних даних. Він використовує очікуваний вихід моделі з попереднього етапу роботи в якості входу, на відміну від згенерованого моделлю. Такий підхід дозволяє тренувати даний тип моделей швидше та надійніше. Саме тому, ми також використовуємо його в роботі.

Найпопулярнішою функцією втрат для моделей мови являється заплутаність(perplexity). У теорії інформації, заплутаність - це міра того, наскільки добре ймовірнісна модель може передбачити зразок. Дана метрика може бути використана для порівняння моделей вірогідності. Низька заплутаність показує, що розподіл ймовірності добре прогнозує вибірку. Заплутаність дискретного розподілу ймовірностей p визначається як на формулі 2.2.

$$e^{H(p)} = e^{-\sum_x p(x)\log(p(x))}, \qquad (2.2)$$

де H(p) - ентропія розподілу, а x приймає значення дискретних подій. Також заплутаність випадкової величини X може бути визначена як заплутаність розподілу за його можливими значеннями x.

Алгоритм оптимізації Адама[25] є розширенням стохастичного градієнтного спуску, який останнім часом отримав широке застосування для глибокого машинного навчання при комп'ютерному зорі та обробці природної мови. Він має такий набір переваг:

- Прямолінійний для реалізації;
- Обчислювально ефективний;
- Невеликі вимоги до пам'яті;
- Інваріантна діагонального маштабування градієнтів;

- Добре підходить для проблем, які є великими з точки зору даних та / або параметрів;
- Підходить для нестаціонарних цільових функцій;
- Підходить для проблем з дуже зашумленими / розрідженими градієнтами.
- Гіперпараметри мають інтуїтивно зрозумілу інтерпретацію і зазвичай вимагають мало налаштування.

Враховуючи дані переваги, було обрано алгоритм оптимізації Адама для тренування рекурентної нейронної мережі, в тому числі і емоційної нейронної діалогової системи.

Отже, фінальне завдання, яке вирішує seq2seq фреймворк це оцінка ймовірності відповіді Y при запиті X, і може бути описане за допомогою формулювання 2.3.

$$P(Y|X) = \prod_{t=1}^{m} P(y_t|y_{<t}, X) = \prod_{t=1}^{m} P(y_t|y_{<t}, x_1, \ldots, x_n), \quad (2.3)$$

де m - довжина відповіді Y = $(y_1, \ldots, y_m)$, n - довжина запиту X = $(x_1, \ldots, x_n)$.

2.2.2 АРХІТЕКТУРА ДІАЛОГОВОЇ СИСТЕМИ

Архітектура розробленої емоційної нейронної діалогової системи досить близька до звичайного Seq2seq підходу, з єдиною відмінністю - для врахування емоційності відповіді на декодер подаються не лише слова, згенеровані на попередньому кроці, а і представлення очікуваної емоції. Саме представлення емоції було отримано аналогічно представленням слів - за допомогою тренування матриці представлень. У нашому випадку існує лише 5 можливих емоцій, тому кількість рядків у матриці представлень емоцій - 5. Щодо кількості стовпців - розмірності вектора емоції - то він був обраний невеликим, адже сама ознака емоції несе менше змісту, ніж слово, яке кодується вектором розмірності 300. Також емоція тексту лежить на вищому рівні абстракції ніж слово, тому і для кодування такої концепції потрібні вектори меншої розмірності. Таким чином кожна емоційна категорія представляється маломірним вектором дійсних чисел. Для кожної емоційної категорії вектор ініціалізується випадково, а потім, через навчання, знаходить правильне місце у просторі емоцій.

Отже, на кожній часовій ітерації на вхід декодеру подаються спільний вектор слова і емоції, об’єднані за допомогою операції конкатенації. Описана архітектура зображена на рисунку 2.4. Таким чином завдання змінюється від описаного в формулі 2.3, до наступного:

$$P(Y|X,e) = \prod_{t=1}^{m} P\left(y_t|y_{<t},X,e\right) = \prod_{t=1}^{m} P\left(y_t|y_{<t},x_1,\ldots,x_n,e\right),$$

де e є {Happiness, Anger, Neutral, Love, Sadness}.

У запропонованій архітектурі емоційної нейронної діалогової системи аналогічно використовується вчительський примус, заплутаність в якості функції втрат та метод оптимізації Адама для тренування ваг мережі.

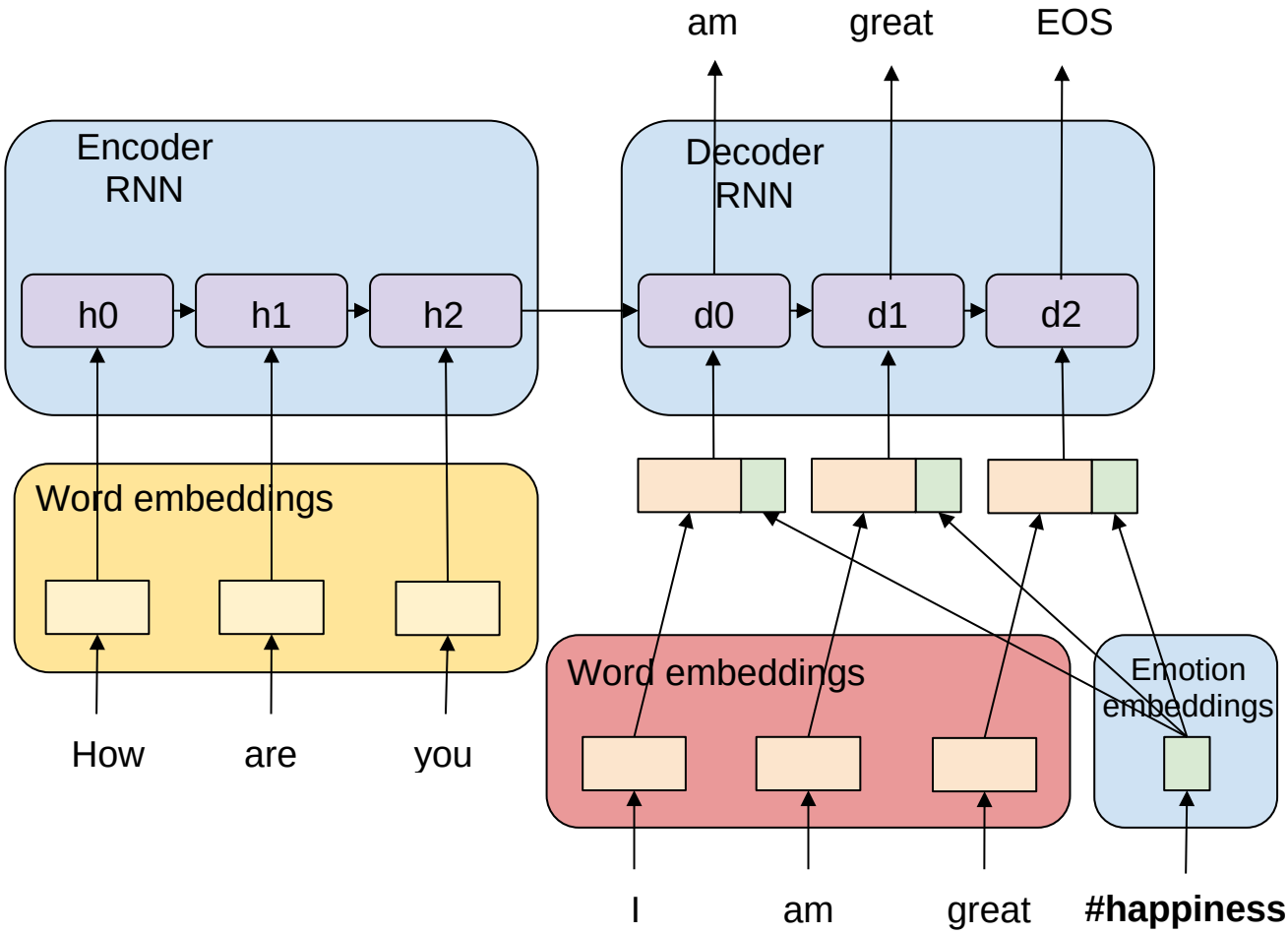


Рисунок 2.4 - Архітектура емоційної нейронної діалогової системи

2.2.3 НАБОРИ ДІАЛОГІВ ДЛЯ ТРЕНУВАННЯ МОДЕЛІ

Для проведення експериментів і тренування ЕНДС було використано 4 різних джерела діалогів. Кожен з наборів даних описаний нижче:

- Cornell Movie Dialogs Corpus[26] - цей корпус містить велику колекцію вигаданих розмов із багатими метаданими, витягнуті з сценаріїв фільмів. Містить в собі 220 579 розмов між 10292 парами персонажів фільмів, включає 9,035 персонажів із 617 фільмів, в цілому 304 713 висловлювань.
- Ubuntu Dialogue Corpus v2.0[27] - набір даних, що містить майже 1 мільйон завершених діалогів, загальною кількістю 7 мільйонів висловів та 100 мільйонів слів. Це унікальний ресурс для досліджень побудови діалогових менеджерів нейронних моделей мови, які можуть використовувати велику кількість неанотованих даних.
- Microsoft Research Social Media Conversation Corpus[28] - колекція з 12676 ідентифікаторів твітів, що представляють 4,232 триступеневих розмовних фрагмента, витягнутих з журналів Twitter. Кожен рядок у наборі даних являє собою єдину трійку контекст-повідомлення-відповідь. Дані були випадково розподілені до тренувального та тестових наборів, що включають 2118 та 2114 трійок відповідно.
- Місячний обсяг коментарів соціальної мережі Reddit[29] - набір даних з графом коментарів користувачів мережі на різну тематику.

Через велику зашумленість та специфічність текстів, а також сленгові висловлювання та неприйнятний для функціональної діалогової системи стиль мовлення набори даних Ubuntu Dialogue Corpus v2.0, Microsoft Research Social Media Conversation Corpus та місячний обсяг коментарів соціальної мережі Reddit не змогли проявити себе, тобто діалогова система на їх основі не змогла навіть на примітивному рівні провести діалог з користувачем, не кажучи вже про використання її для пошуку курсів. Саме тому у фінальному варіанті системи тренувальним набором був лише Cornell Movie Dialogs Corpus. Це вирішує низку інших проблем: разом всі корпуси займають 40.5 Гб дискового місця і можуть бути класифіковані як великий масив даних і відповідно вимагають використання інших рішень і підходів; словник найчастіших слів у корпусі на базі субтитрів набагато чистіший і легше сприймається пересічному користувачу, на відміну від словника, який тренувався на базі форумів

операційної системи Linux. Після автоматичної анотації емоцій у корпусі за допомогою класифікатора FastText було отримано показаний на рисунку 2.5 розподіл.

Value	Count	Frequency (%)
neutral	75129	33.9%
anger	72310	32.6%
happiness	48122	21.7%
sadness	17951	8.1%
love	8105	3.7%

Рисунок 2.5 - Розподіл автоматично анотованих емоцій у Cornell Movie Dialogs Corpus

2.3 ДОПОВНЕННЯ СИСТЕМИ ЗНАННЯМИ

2.3.1 РЕКОМЕНДАЦІЯ ОНЛАЙН КУРСІВ

Оскільки пошук інформації є ключовою діяльністю в будь-якому виді навчальної системи, створення рекомендаційних систем привертає все більшу вагу [30]. Проте рекомендація онлайн курсів - не просте завдання через низку причин:

- педагогічні принципи: мотивація та критика є індивідуальними для кожного учня;
- передумови: існують залежності - деякі курси не можуть бути розпочаті до закінчення попереднього;
- результати навчання не можуть бути виміряні.

Порівнюючи типову рекомендаційну систему для продажу певних продуктів до навчальної системи з такою технологією, варто зазначити, що остання має довгостроковий показник ефективності: знання можна виміряти успішною зайнятістю або їх використанням на практиці, а продажі мають ряд чітко визначених короткострокових показників, такі як прибуток або кількість кліків у певному вікні.

Більшість із цих проблем можна уникнути припущенням, що курси з семантично подібним матеріалом надають подібний ступінь освітнього впливу на користувача та можуть доповнювати один одного. За таким припущенням ми можемо побудувати рекомендаційну модель, яка дозволить запитувати семантично подібні

курси та об'єднати такі курси в програму. Все це можна досягти шляхом кодування курсів у векторному просторі через їх текстове подання та матеріали.

2.3.2 НАБІР ДАНИХ ПРО ОНЛАЙН КУРСИ

Набір даних нашого дослідження складався з опису 4066 онлайн курсів від найпопулярніших постачальників MOOC, включаючи Coursera, edX та Udacity, розподілених серед 40 категорій. Ці дані включають назву, дату початку, установу організації (університет, компанію), платформу постачальника, мову, категорію, середню оцінку та текстовий опис. Однак для створення векторної моделі було використано лише текстові елементи, зокрема назву та опис. Проте інформацію, що залишилася, можна об'єднати з векторною моделлю для подальших спроб вдосконалити модель рекомендації.

Текстовий опис курсу був отриманий шляхом об'єднання його назви, короткого опису та детального опису. Для створення векторної моделі був обраний підхід Google Word2Vec[31] через наявність 3 мільйонів 300-мірних представлень слів та фраз, попередньо натренованих на основі набору даних Google News, розміром 100 мільярдів слів. Для кожного слова в текстовому описі певного курсу було вилучено відповідний 300-мірний вектор. Потім дані вектори були усереднені по кожній осі для генерації 300-мірного вектору курсу, як показано на рисунку 2.6.

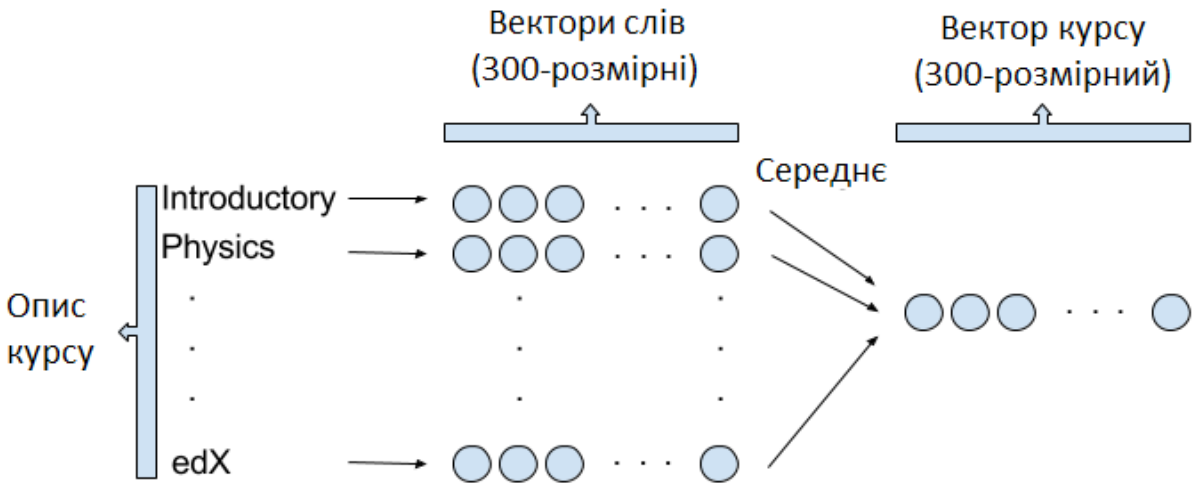


Рисунок 2.6 - Побудова представлення онлайн курсу

Косинусова відстань була обрана як міра відстані для векторів курсів. Щоб побудувати інтуїцію та виявити приховані тенденції щодо остаточного векторного простору, корисно побудувати візуалізацію. Проблема з цим полягає в тому, що вектори є 300-мірними і не можуть бути візуалізовані за допомогою звичайних методів. Проте існує низка методів зменшення розмірності, таких як метод головних компонентів (МГК, PCA) [32] та t-розподілене стохастичне сусіднє представлення (t-SNE) [33], які можуть бути використані для проектування наших векторів у простір меншої розмірності. t-SNE найкраще працює з векторами слів, оскільки він фіксує структуру подібності у всьому просторі. Саме тому його було обрано для візуалізації векторів курсів. На рисунку 2.7 можна побачити візуалізацію частини простору курсів, а саме категорії “Computer Science”. Видно, що цей простір має семантичну структуру і схожі за значенням курси групуються в кластери, наприклад “Creating Virtual Reality” і “Computer Graphics” чи “Python Data Visualization” і “Python Data Representations”.

Рисунок 2.7 - t-SNE візуалізація простору онлайн курсів у категорії “Computer Science”

2.3.3 ГЕНЕРАЦІЯ ВІДПОВІДЕЙ

При такому підході виникає інша проблема - якщо кількість курсів для пошуку збільшується, то і час на рекомендацію збільшується, адже запит потрібно порівняти

з великою кількістю багатовимірних векторів. У такому разі для покращення швидкодії системи важливо погрупувати чи прокластеризувати курси. Так як у нашому наборі даних було поле ‘категорія’, необхідність кластеризації відпала, а у архітектурі системи відбулись наступні зміни: замість того, щоб при кожному запиті шукати найближчі курси у всьому просторі курсів, спочатку проводиться пошук найближчої категорії, а після того - найближчого курсу в знайденій категорії.

Після знаходження семантично найближчого до запиту курсу, його назва та посилання конкатенуються із випадковим шаблоном зі списку шаблонів, створеного користувачем/експертом. Такий список може включати як прості вислови “Try”, так і більш складні “Why don’t you try” для забезпечення диверсифікації відповідей-рекомендацій.

Для того, щоб система розуміла в якому випадку потрібно генерувати власну відповідь з нейронної рекурентної мережі, а в якому потрібно порекомендувати курс, було введено поріг близькості до категорії та курсу. Таким чином, якщо при пошуку найближчої категорії жодна з них не досягає заданого порогу, система переключається на генерацію відповіді з мережі.

2.4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.4.1 ВИБІР ПРОГРАМНОГО СЕРЕДОВИЩА РОЗРОБКИ

Для програмної реалізації описаних методів та алгоритмів було обрано мову програмування Python версії 3.6.3. Python підтримує кілька парадигм програмування, в тому числі об'єктно-орієнтованого, імперативного та функціонального програмування або процедурних стилів. Вона має динамічну систему типів і автоматичне керування пам'яттю, а також має велику кількість бібліотек різного призначення. Його філософія дизайну підкреслює читаність коду, а його синтаксис дозволяє програмістам висловлювати поняття в меншій кількості рядків коду, ніж це можливо в таких мовах, як C ++ або Java. Python інтерпретатори доступні для багатьох операційних систем, дозволяючи запускати код написаний на Python на найрізноманітніших системах. За допомогою сторонніх інструментів Python код може

бути упакований в автономні виконувані програми для деяких з найбільш популярних операційних систем, так що програмне забезпечення Python може бути поширеним і використовуватись у різних середовищах, навіть там де інтерпретатор Python не встановлений. Замість того, щоб вимагати всіх функцій у ядрі мови, Python був розроблений, щоб бути максимально розширюваним. Основою реалізації класифікатора емоцій стала бібліотека scikit-learn[34] – відкрита безкоштовна бібліотека для машинного навчання, у якій зібрані алгоритми для вирішення різних задач – класифікації, прогнозування, регресії чи кластеризації, а також готові скрипти FastText, які запускались автоматично з Python програми.

Так як у основі ЕНДС лежить такий тип методів як глибинне навчання, то важливо було обрати фреймворк для розробки серед кількох відомих альтернатив, а саме: TensorFlow[35], PyTorch[36] та інших. Для реалізації діалогової системи у даній роботі було вибрано PyTorch через наступні його переваги:

- надає інтерфейс тензорів, які можуть працювати як на центральному процесорі, так і на графічному процесорі, таким чином значно прискорюючи обчислення;
- пропонує широкий спектр тензорних процедур для прискорення обчислень і інструментів для зручності розробки наукових застосувань, таких як нарізка, індексація, математичні операції, лінійна алгебра;
- має унікальний спосіб створення нейронних мереж - методика, що називається автоматичною диференціацією у оберненому режимі, яка дозволяє змінювати, як працює мережа з нульовими затримками чи накладними витратами;
- не є просто Python зв’язкою до монолітної системи на C++. Вона побудована так, щоб бути глибоко інтегрованою в Python. Можна написати нові шари нейронної мережі на тому ж Python, використовуючи улюблені бібліотеки.

Для побудови мінімального веб сервісу з програмним інтерфейсом було обрано Bottle - швидкий, простий і легкий у використанні WSGI мікро-фреймворк для Python.

Його основна перевага в поширенні як єдиний файловий модуль і не використанні інших залежностей, крім стандартної бібліотеки Python. Саме тому він ідеально підходить для створення мінімального API.

На клієнтській частині системи використовується Angular - платформа для створення веб-застосунків на базі open-source на основі TypeScript. Через свою модульність вона дозволила елегантно імплементувати користувацький інтерфейс чат-боту.

Лістинг ЕНДС відображений у додатку А.

2.4.2 АПАРАТНІ ВИМОГИ ДО ЕКСПЕРИМЕНТІВ

Висока розмірність нейронних мереж призводить до збільшення обчислень і руху даних. Вища розмірність та збільшення кількості отриманих даних і обчислень випливає з факту, що ваги мережі також повинні бути прочитані і зберігатись. Врешті решт, навчання вимагає значної кількості анотованих даних (особливо для глибинного навчання), а також численних обчислень ітерації зворотного поширення помилки для визначення значення ваг мережі[37]. Для пришвидшення даних процесів ефективно використовувати графічні процесори.

Еволюцію графічних процесів зумовлюють постійно зростаючі вимоги комп'ютерних графічних додатків, зокрема ігор. Комп'ютерна графіка побудована на основі лінійної алгебри з плаваючою точкою, що означає, що графічні процесори дуже ефективні при матричних операціях. Підготовка глибинних нейронних мереж зазвичай виражається як величезні обчислювальні графи, де вузли є операціями, а ребра представляють потоки даних. Ці вузли, як правило, є матричними операціями - операції лінійної алгебри або поелементні операції.

Саме тому дана робота для функціонування за прийнятний час вимагає наявність сучасного графічного процесора. Зазначимо, що це стосується лише частини роботи зв’язаної з рекурентними нейронними мережами, а отже побудови ЕНДС. У дослідженні було використано відеокарту GeForce GTX 1080 Ti, яка дозволила тренувати діалогову систему до прийнятного результату за приблизно 60 годин.

Досліди з тренуванням класифікатора емоцій вимагали підбору гіперпараметрів за допомогою сіткового пошуку, а даний процес досить прямолінійно проводити паралельно, запускаючи моделі з різними гіперпараметрами в різних процесах. Для покращення швидкодії і паралелізації даної процедури, було використано процессор Intel Core i7-6850K з 6 ядрами в 2 потоки, тобто фактично можна було тренувати одночасно 12 моделей.

2.5 ВИСНОВОК

У розділі було описано 3 основних компонента розробки емоційної нейронної діалогової системи для пошуку онлайн курсів, а саме: класифікатора емоцій, рекурентної нейронної мережі та модуля рекомендації курсів. Проведені експерименти показали, що серед логістичної регресії, випадкових лісів, дерев рішень та FastText найкращим для класифікації емоцій виявився підхід FastText.

Щодо рекурентної нейронної мережі, то була розроблена архітектура, яка б враховувала емоцію відповіді, таким чином надаючи змогу користувачу контролювати емоційне забарвлення діалогу. Основна ідея архітектури - врахування представлень емоцій, які тренуються одночасно з вагами мережі. Такі представлення конкатенуються з представленнями слів перед етапом декодування та передаються на вхід декодеру.

Модуль рекомендації онлайн курсів був спроектований так, щоб надавати семантично найближчі рекомендації до запиту користувача, при цьому враховуючи обчислювальне навантаження. Для пошуку онлайн курсу запит користувача спочатку буде семантично співставлятись зі списком категорій, а згодом з курсами у найближчій категорії. Таким чином будуть знаходитись найрелевантніші відповіді на запит користувача за значно менший час ніж повний перебір.

Для створення можливості відтворити дане дослідження були описані програмна реалізація та апаратні вимоги до системи, а також деталі фаз тренування класифікатора та нейронної мережі, а також описано систему генерації відповідей при пошуці курсів.

3 ОГЛЯД РЕЗУЛЬТАТІВ РОЗРОБКИ СИСТЕМИ

3.1 КІЛЬКІСНІ РЕЗУЛЬТАТИ

Кількісні результати тренування емоційної нейронної діалогової системи виражаються у значеннях заплутаності при генерації відповіді у порівнянні з базовим неемоційним seq2seq підходом. Отримані значення заплутаності зазначені у таблиці 3.1. Як і очікувалось, заплутаність емоційної нейронної діалогової системи менша, ніж базового підходу, що означає, що мережа обирає з меншої кількості варіантів при генерації наступного слова. Скоріше за все це пов’язано з тим фактом, що вона має доступ до додаткових даних у вигляді емоції, а отже область пошуку зменшується. З іншого боку, це може свідчити про перенавчання системи, адже вона могла запам’ятати типові для певної емоції слова і генерувати її з більшою частотою, зменшуючи заплутаність.

Таблиця 3.1 – Заплутаність базового та емоційного підходів

Підхід	Заплутаність
Seq2seq	5.2
ЕНДС	4.85

Варто зазначити, що метрики для оцінювання діалогових систем часто не відображають задоволення від користування системою. Так наприклад у статті [38] автори виявили, що всі показаники, які використовуються у літературі для оцінювання діалогових систем, мають слабку чи нульову кореляцію з людськими судженнями. Саме тому єдиним правильним рішенням було б опитати велику кількість користувачів і зробити висновки. На жаль, такий процес надто часо- та ресурсоємкий і не міг бути проведений у даній роботі.

3.2 ЯКІСНІ РЕЗУЛЬТАТИ

Щоб краще зрозуміти якість сформованих відповідей зі сторони вмісту та емоцій, була виконана ручна оцінка. Враховуючи запит та емоцію відповіді, було проведено візуальний аналіз результату відпрацювання емоційної нейронної

діалогової системи. Кілька прикладів таких запитів та відповідей продемонстровано у таблицях 3.2 та 3.3.

Як бачимо, відповіді з емоцією ‘Neutral’ мають схожий зміст до звичайної Seq2seq моделі, інші ж емоційні відтінки різняться за текстом відповідно до заданої категорії. Хоча у більшості випадків діалогова система відповідає з бажаним емоційним забарвленням, через те, що прикладів категорій ‘Sadness’ та ‘Love’ в наборі даних для тренування було порівняно мало, відповіді з ними часто схожі на seq2seq або ‘Neutral’.

Таким чином дана система здатна генерувати відповіді, які залежать від заданого емоційного забарвлення, в той час як звичайний seq2seq генерує відповіді з випадковим настроєм.

Таблиця 3.2 – Згенеровані відповіді базового та емоційного підходів на запит

Учасник діалогу	Емоція	Текст
Користувач	-	How are you?
Seq2seq	-	Pretty good.
ЕНДС	Happiness	Very good.
ЕНДС	Anger	Oh my god.
ЕНДС	Neutral	I am fine.
ЕНДС	Sadness	Fine, thanks.
ЕНДС	Love	Good, good.

Таблиця 3.3 – Згенеровані відповіді базового та емоційного підходів на запит

Учасник діалогу	Емоція	Текст
Користувач	-	What do you want to do?
Seq2seq	-	I do not know.
ЕНДС	Happiness	I want to talk to you about this.
ЕНДС	Anger	I do not.

ЕНДС	Neutral	I do not know.
ЕНДС	Sadness	I can not.
ЕНДС	Love	I want you to believe me.

3.3 ПОШУК ОНЛАЙН КУРСІВ ЗА ДОПОМОГОЮ СИСТЕМИ

Найбільш важливим параметром у системі під час пошуку онлайн курсів є поріг семантичної віддаленості від запиту. Якщо поріг поставити надто високим, система почне рекомендувати курси коли це не вимагається користувачем, наприклад при звичайному питанні: “What music do you like?”. Це пов’язано, з тим, що косинусова відстань до курсів на музичну тему буде нижчою за поставлений високий поріг через наявність слова “music” у запиті. У цей же час, якщо поставити надто низький поріг, лише дуже семантично близькі запити будуть ініціювати процедуру пошуку курсів. Таким чином, користувачу буде важно знайти правильне формулювання запиту у діалозі. Саме тому, дуже важливо знайти оптимальний поріг для прийнятного функціонування системи. У експериментах було виявлено, що поріг у 0.8 дозволяє рекомендувати курси саме тоді коли це потрібно. Приклад користувацького діалогу наведений на рисунку 3.1, де синім позначені запити(слово після знаку # - бажана емоція відповіді), а сірим відповіді системи.

3.4 ВИСНОВОК

Результати показують, що гібридний підхід до створення діалогових систем для конкретних завдань може ефективно працювати завдяки двом компонентам: емоційній нейронній діалоговій системі та пошуковому модулі, заснованому на семантичній близькості. При цьому у даному дослідженні вдалось покращити метрику ефективності діалогової системи - заплутаність - відносно базового варіанту seq2seq на 6-7% за допомогою вводу концепції емоцій.

Якісні результати продемонстрували, що ЕНДС має високорівненве розуміння емоцій і здатна генерувати відповіді в залежності від заданої категорії. Таким чином

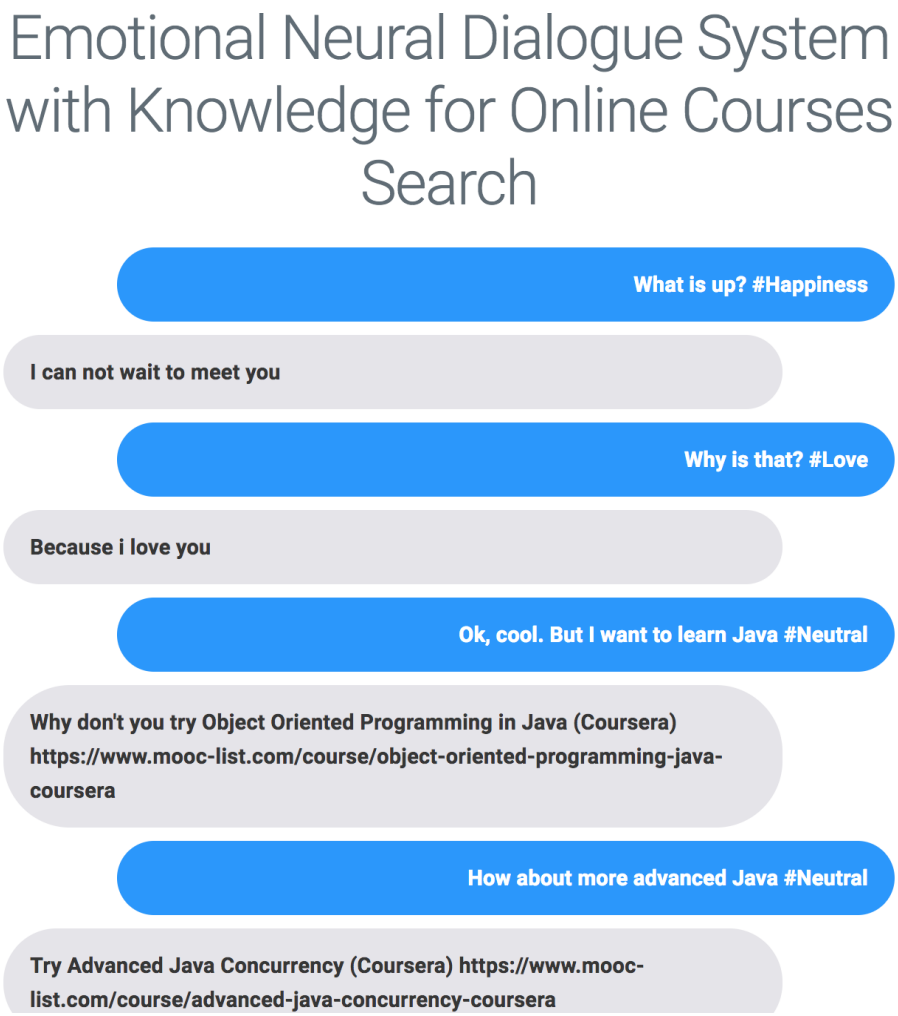


Рисунок 3.1 - Приклад діалогу з ЕНДС та рекомендації курсів у розробленому користувацькому інтерфейсі

збільшується емоційний інтелект системи і вона може більше симпатизувати користувачам у повсякденному спілкуванні.

Рекомендація онлайн курсів також проходить у задуманий спосіб, система автоматично розуміє коли користувач бажає знайти курс для навчання, а коли спілкується на довільну тему. У останньому підрозділі було наведено рекомендацію по підбору порогу віддаленості при семантичному пошуці курсів для максимально безшовної комунікації з користувачем.

4 РОЗРОБКА СТАРТАП-ПРОЕКТУ «ЕМОЦІЙНА ДІАЛОГОВА СИСТЕМА ДЛЯ РЕКОМЕНДАЦІЇ ОНЛАЙН КУРСІВ»

4.1 ОПИС ІДЕЇ ПРОЕКТУ

Описана технологія може буди реалізована в якості веб-додатку у вигляді текстового мессенджера для зручного спілкування з чат-ботом та рекомендації курсів користувачу. Цей розділ ставить перед собою мету реалізації технології емоційної нейронної діалогової системи зі знаннями для рекомендації онлайн курсів, а також проведення маркетингового аналізу та виявлення ринкових можливостей використання роботи. Процес включає в себе:

- імплементацію веб-сервісу з використанням підходів та технологій описаних у попередніх розділах роботи;
- розробку стратегії виходу конкурентоспроможного продукту на ринок та подальший розвиток стартапу.

Для отримання цілісного уявлення про зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів важливо побудувати таблицю з описом ідеї можливими напрямками застосування основними вигодами, що може отримати користувач товару.

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення веб-додатку у формі мессенджера для швидкого та комфортного пошуку онлайн курсів за допомогою текстового спілкування з нейронною діалоговою системою.	1. Пошук онлайн курсів на задану тематику.	Користувач буде мати змогу швидко знайти релевантний курс для вивчення, не виходячи з улюбленого мессенджера.
	2. Вивчення англійської мови через спілкування з чат-ботом.	Користувач буде мати змогу покращити рівень своєї письмової англійської через постійну практику з

		діалоговою системою.

Отже, ідея проекту заключається в тому, що через веб-додаток у формі месенджера користувач зможе шукати масові відкриті онлайн курси, а також покращувати свої навички спілкування та письма англійською мовою. При цьому в основі додатку лежить емоційна нейронна діалогова система, яка здатна буде краще імпонувати користувачу.

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності. Тому у таблиці 4.2 показано чим товар відрізняється від існуючих аналогів чи замінників на основі техніко-економічних властивостей та характеристик.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1	Конкурент 2	Конкурент 3			
1.	Форма виконання	Веб-сервіс +чат-бот	Веб-додаток	Веб-додаток	Веб-сервіс			+
2.	Собівартість	Низька	Низька	Низька	Висока		+	
3.	Емоційність	Контрольована	Випадкова	Випадкова	Випадкова			+

4.	Крос-платформенність	Так	Так	Ні	Так			+
5.	Потреба в інтернеті	Так	Так	Ні	Так		+	

Сильними сторонами даного проекту є те, що форма виконання ідеї окрім веб-сервісу включає інтерфейс чат-боту, тому що така форма зручніша для користувачів, емоційність такого чат-боту, тому що це дозволяє збільшити задоволення користувача при експлуатації, а також крос-платформенність, адже здатність підтримувати велику кількість платформ напряму збільшує кількість потенційних клієнтів. Всі інші характеристики є нейтральними. Тому даний проект можна вважати конкурентоспроможним.

4.2 ТЕХНОЛОГІЧНИЙ АУДИТ ІДЕЇ ПРОЕКТУ

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Створення емоційної нейронної діалогової системи	PyTorch	Наявна	Безкоштовна, доступна
		Tensorflow	Наявна	Безкоштовна, доступна
		Wolfram Mathematica	Наявна	Приватна, недоступна
Обрана технологія реалізації ідеї проекту: для створення емоційної нейронної діалогової системи обрана технологія PyTorch, яка є безкоштовною та зручною для імплементації				

глибоких нейронних мереж.

Отже, проект буде реалізовано за допомогою технології PyTorch, адже вона зручна та дозволяє писати імперативний код, на відміну від Tensorflow, а також безкоштовна і доступна на ринку, на відміну від Wolfram Mathematica. Тому даний проект можна вважати технологічно здійсненним.

4.3 АНАЛІЗ РИНКОВИХ МОЖЛИВОСТЕЙ

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводимо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	100
2	Загальний обсяг продаж, грн/ум.од	1000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу	Висока вартість початкового капіталу
5	Специфічні вимоги до стандартизації та сертифікації	-
6	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Отже, середня норма рентабельності в галузі менша, ніж банківський відсоток на вкладення. Тому має сенс вкласти кошти в саме цей проект, адже проект не має специфічних вимог до стандартизації та сертифікації, бо він буде виконаний у вигляді мобільного додатку і буде розміщений в App Store з необмеженим доступом.

Далі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Автоматизація спілкування з клієнтами	а) Компанії, яким потрібно автоматизувати відділ підтримки; б) компанії, яким потрібно автоматизувати відділ продажів/прийняття замовлень	а) націлені на обмежений список запитань-відповідей, коротке спілкування з клієнтом б) націлені на постійне поповнення різноманіття послуг, середнє спілкування з клієнтом	Чат-боти: Надійні, Функціональні, Симпатизують клієнтам споживачів; Постачальник чат-ботів: Надає підтримку, Постійно покращує рішення

Визначено характеристики стартап-проекту: основну потребу, що формує ринок - автоматизацію спілкування з клієнтами; наведено основні цільові сегменти ринку – компанії з великими відділами підтримки, продажів, замовлень тощо; відмінності у поведінці різних потенційних цільових груп клієнтів - отримання інформації та різноманітне спілкування; затверджено основні вимоги до споживачів – надійні та високофункціональні чат-боти з яскраво вираженою емоційною складовою.

Далі складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. 4.6-4.7).

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Зростаюча вимогливість покупців	Споживачі в час технологічних іновацій та розвитку обробки природньої мови вимагають найбільш розвинених чат-ботів	Створення дослідницької лабораторії, присвяченої розробці найкращих технологій у сфері чат-ботів
2	Зміна потреб і смаків покупців	Окрім служб підтрики/продажу клієнти можуть придумати інші застосування, потрібні на тому чи іншому етапі розвитку клієнтського продукту	Додавати нові проекти і види чат-ботів до списку послуг
3.	Збільшення витрат на технічну підтримку	Невчасне реагування на сучасний ринок потреб користувачів	Вчасно оновлювати програмне забезпечення
4.	Конкуренція	Вихід на ринок великої	вихід з ринку;

		компанії	запропонувати великій компанії поглинути себе; передбачити додаткові переваги власного сервісу для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
5.	Зменшення кількості замовників	Зменшення зацікавленості замовників у наданих сервісах	Постійні оновлення алгоритму

Було наведено основні фактори загроз стартап-проекту. Найбільшою загрозою для проекту є зміна потреб користувачів (користувачам необхідні нові види чат-ботів) та їх вимогливість (користувачам потрібні найбільш реалістичні чат-боти з новими технологіями). Для зменшення цих загроз потрібне створення дослідницької лабораторії, присвяченої розробці найкращих технологій у сфері чат-ботів, а також передбачення можливості диференціації призначень наданих сервісів і своєчасного оновлення програмного забезпечення і користувацького інтерфейсу. Найменшими загрозами є збільшення витрат на технічну підтримку та зменшення кількості користувачів замовників.

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Підвищились технологічні бар’єри входу на ринок: розробка чат-ботів - важкий дослідницький проект	Для реалізації функціональності, задовільної для користувачів,	Наймати та навчати кадри, які могли б революціонувати область

		потрібні найкращі дослідники	
2	Можливість швидкого розвитку у зв'язку з різким зростанням попиту на ринку	Майже кожна інтернет-компанія хотіла б автоматизувати спілкування з клієнтами, тому користувачів багато	Максимальне захоплення ринку за рахунок впливових клієнтів
3	Зростання можливостей потенційних покупців	Зростання фінансування досліджень у галузі машинного навчання	Запропонувати свої послуги зацікавленим підприємствам
4	Зниження довіри до конкурента 1	У додатку конкурента 1 нещодавно було знайдено витік інформації, яка збиралася для аналітики	При виході на ринок звертати увагу покупців на безпеку нашого додатку
5	Зменшення витрат на технічну підтримку	Збільшення продуктивності роботи штату компанії за рахунок підвищення їхнього професійного рівня	Підвищувати рівень кваліфікації своїх співробітників

Було наведено основні фактори сприяння ринковому впровадженню проекту: підвищення технологічних бар’єрів входу на ринок: розробка чат-ботів - важкий дослідницький проект, можливість швидкого розвитку у зв'язку з різким зростанням попиту на ринку, зростання можливостей потенційних покупців. Основними реакціями компанії є: надання своїх послуг зацікавленим підприємствам, безпечність

програмного забезпечення, найм та навчання кадрів, які могли б революціонувати область та максимальне захоплення ринку за рахунок впливових клієнтів.

Надалі визначаються загальні риси конкуренції на ринку (табл. 4.8).

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції - олігополія	Існує мала кількість фірм-конкурентів на ринку	Створення більш технологічно досконалих рішень, ніж конкуренти
2. За рівнем конкурентної - інтернаціональна	Майже всі конкуренти - закордонні	Можна знайти дешевших дослідників
3. За галузевою ознакою - внутрішньогалузева	Конкуренти мають сервіси, які використовуються лише всередині даної галузі	Добре описані рамки галузі та способи впливу на неї
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме - програмне забезпечення	Конкуренція між різними видами чат-ботів
5. За характером конкурентних переваг - нецінова	Вдосконалення технології створення дотатку, щоб собівартість була нижчою	Поліпшення якості продукції
6. За інтенсивністю - не марочна	Бренди відсутні	Легше вийти на ринок молодій компанії

Було наведено проведено аналіз конкуренції на ринку, а саме визначено: тип конкуренції - олігополія; конкуренція за рівнем конкурентної боротьби - міжнародна; конкуренція за галузевою ознакою - внутрішньогалузева; конкуренція за видами

товарів - товарно-видова; конкуренція а характером конкурентних переваг - нецінова; конкуренція за інтенсивністю - не марочна. Також було наведено можливі дії компанії, щоб бути конкурентноспроможною: створення більш технологічно досконалих рішень, ніж конкуренти; пошук дешевших дослідників; використання способів впливу на галузь; створення нових видів чат-ботів; поліпшення якості продукції.

Далі розробляється перелік факторів конкурентоспроможності для ринку на основі аналізу складових моделі 5 сил М. Портера (табл. 4.9).

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Google IBM	Бар'єр - потужний дослідницький відділ	Amazon Google Сильні сторони: стабільні, мають підтримку	Rozetka eBay	-
Висновки:	Дуже інтенсивна боротьба за першість в чат-бот технологіях	Потенційно може вийти будь-яка компанія з необхідними ресурсами	Нічого не диктують, в крайньому разі можна і самому серверів накупити	Диктують умови, але у більшості вони уніфіковані - чат-бот-підтримки, чат-бот-продажник	-

Отже, з огляду на конкуренту ситуацію можна з впевненістю сказати, що проект має можливість роботи на ринку, тому що серед наведених конкурентів немає тих, які б могли його потіснити, адже розроблене рішення спрощує та пришвидшує роботу спеціаліста. Можна виділити основні сильні сторони продукту, які б допомогли стати конкурентоспроможним на ринку - надійність, а також технологія емоційності в чат-ботах.

На основі аналізу висновків, наведених вище, визначається та обґрунтовується перелік факторів конкурентоспроможності (табл. 4.10).

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Науково-технічний потенціал	Кращі дослідження=вища якість послуг
2	Кадровий потенціал	Кращі кадри=кращі дослідження

Було наведено основні фактори конкурентноспроможності, які будуть представлені на ринку, а саме: науково-технічний потенціал, який дозволяє надавати вищу якість послуг, а також кадровий потенціал, який гарантує швидші та якісніші дослідження у галузі чат-ботів.

За визначеними факторами конкурентоспроможності проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін «Емоційна діалогова система для рекомендації онлайн курсів»

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з «Емоційні чат-боти»						
			-3	-2	-1	0	1	2	3

1	Науково-технічний потенціал	19			+				
2	Кадровий потенціал	19			+				

Було наведено порівняльний аналіз сильних сторін проекту товарів-конкурентів і нашого підприємства. Найбільше балів набрано для таких факторів конкурентноспроможностей – науково-технічний потенціал та кадровий потенціал.

Далі проводимо SWOT-аналіз (табл. 4.12).

Таблиця 4.12 – SWOT- аналіз стартап-проекту

Сильні сторони: найкращі фахівці в області, унікальний алгоритм емоційності чат-ботів	Слабкі сторони: слабкий маркетинговий відділ, відсутність репутації, молода компанія
Можливості: стрімкий ріст ринку та технологічності рішень у сфері чат-ботів	Загрози: конкуренти значно більш відомі і потужніші, мають більше ресурсів, з часом здатні придумати кращий алгоритм або покращити наш

Отже, внутрішні можливості компанії і спроможності щодо виведення продукту на ринок характеризуються такими сильними і слабкими сторонами: сильні - найкращі фахівці в області, унікальний алгоритм емоційності чат-ботів; слабкі - слабкий маркетинговий відділ, відсутність репутації, молода компанія. Ринкові та можливості компанії щодо зовнішнього оточення характеризуються можливостями і загрозами: можливості - стрімкий ріст ринку та технологічності рішень у сфері чат-ботів; загрози - конкуренти значно більш відомі і потужніші, мають більше ресурсів, з часом здатні придумати кращий алгоритм або покращити наш.

На основі SWOT-аналізу складаються альтернативи ринкового впровадження стартап-проекту (табл. 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Покращуємо алгоритм настільки, щоб конкурентів залишити далеко позаду	70%	1 рік
2	Виходимо на ринок з беземоційними чат-ботами, граємо на швидкому рості ринку	50%	6 місяців

Отже, було визначено альтернативу ринкового впровадження стартап-проекту - покращення алгоритму настільки, щоб конкурентів залишити далеко позаду, адже для неї отримання ресурсів є більш ймовірним, а строки реалізації не значно більші, ніж альтернатива.

4.4 РОЗРОБКА РИНКОВОЇ СТРАТЕГІЇ ПРОЕКТУ

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних Клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Компанії, яким потрібно автоматизувати	Готові, якнайшвидше	Попит - високий	Не дуже інтенсивна	Середня

	відділ підтримки;				
2	Компанії, яким потрібно автоматизувати відділ продажів/прийняття замовлень	Готові, якнайшвидше	Попит - середній, через недовіру технологіям	Не дуже інтенсивна	Середня
Які цільові групи обрано: обрано обидві групи, так як вони майже не відрізняються, а проект їх може задовольнити					

Отже, було вибрано основні цільові групи: компанії, які потребують автоматизацію відділу підтримки та компанії, які потребують автоматизацію відділу продажів чи прийняття замовлень, адже обидві групи мають схожі вимоги та готові сприйняти продукт.

Далі визначається базова стратегія розвитку (табл. 4.15).

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Створення чат-ботів - емоційних, що є новою і унікальною властивістю	Створення продуктових новацій та їх маркетинг	Основна позиція - емоційність, адже є ключовою технологією і перевагою нашого	Стратегія диференціації

			проекту	

Отже, була обрана альтернатива розвитку проекту - створення чат-ботів - емоційних, що є новою і унікальною властивістю. Визначена базова стратегія розвитку - диференціація, інструментом реалізації якої є створення продуктових новацій та їх маркетинг, адже емоційність чат-ботів є ключовою технологією і перевагою нашого проекту.

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.16).

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Емоційність - так, чат-боти для автоматизації - ні	Шукати нових і забирати існуючих	Стабільність/надійність сервісів, підтримка, інтерфейс	Оборонна

Отже, було визначено базову стратегію конкурентної поведінки - оборонну, адже емоційність чат-ботів у автоматизації є унікальною характеристикою продукту. Також матиме технічну підтримку та зручний інтерфейс, що формуватиме довіру і прихильність споживачів.

Далі визначається стратегія позиціонування проекту, яка допоможе користувачам ідентифікувати програмний продукт (табл. 4.17).

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару	Базова стратегія	Ключові конкурентоспромож	Вибір асоціацій, які мають

	цільової аудиторії	розвитку	ні позиції власного стартап-проекту	сформувати комплексну позицію власного проекту (три ключових)
1	Коректна робота, надійність, підтримка	Стратегія диференціації	Основна позиція - емоційність, адже є ключовою технологією і перевагою нашого проекту	Швидкодія, безпека, простота

Отже, було визначено стратегію позиціонування, а саме визначено основні вимоги до товару цільової аудиторії: коректна робота, надійність, підтримка; базову стратегію розвитку: диференціація; ключові конкурентоспроможні позиції стартап-проекту: емоційність, адже є ключовою технологією і перевагою нашого проекту. Також сформовано комплексну позицію проекту: швидкодія, безпека, простота.

4.5 РОЗРОБКА МАРКЕТИНГОВОЇ ПРОГРАМИ

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Автоматизація певного бізнес-процесу: користувацької підтримки та відділу продажів	Не потрібно наймати додаткових фахівців для консультацій користувачів	Емоційність чат-ботів робить спілкування з автоматизованою системою значно приємнішим

Надалі розробляється трирівнева маркетингова модель товару (табл. 4.19).

Таблиця 4.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Автоматизація певного бізнес-процесу: користувацької підтримки та відділу продажів. Внаслідок цього не потрібно наймати додаткових фахівців для консультацій користувачів. Також це допомагає швидше та ефективніше працювати з клієнтами, але для цього потрібно максимально приємний користувачам чат-бот.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Надійність
2. Емоційність
3. Найдовший час відповіді</td><td>М
М
М</td><td>Тл
Тл
Тл</td></tr>
<tr><td colspan="3">Відповідає нормам розмовної поведінки, протестовано на реальних клієнтських діалогах</td></tr>
<tr><td colspan="3">Веб-сервіс з відкритим інтерфейсом, до якого звертаються клієнтські додатки</td></tr>
<tr><td colspan="3">EmoBotCompany: EmoBot</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">Акції, знижки</td></tr>
<tr><td colspan="3">Підтримка</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: патент.</td></tr>
</table>

Отже, було описано три рівня моделі товару: товар буде мати вигляд чат-боту, який автоматизуватиме користувацькі бізнес-процеси, а саме користувацької підтримки та відділу продажів. Його основні характеристики: емоційність, надійність

та найдовший час відповіді. Технологія буде захищена від копіювання за рахунок патенту.

Наступним кроком є визначення цінових меж (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	125 грн. / 1000 викликів	250 грн. / 1000 викликів	>100000 грн. / місяць	275 грн. – 500 грн. / 1000 викликів

Визначено межі встановлення ціни на мобільний додаток, а саме рівень цін на товари-замінники – 125 грн. за 1000 викликів програмного інтерфейсу, рівень цін на товари-аналоги 250 грн., рівень доходів цільової групи споживачів - 100000 грн, верхня та нижня межі встановлення ціни на товар – 275-500 грн. Аналіз був проведений експертним методом.

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21).

Таблиця 4.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Клієнти зазвичай оцінять можливості платформи через безкоштовну	Фінансування витрат на функціонування	Канал нульового рівня	Проводити збут власними силами

	пробну версію, а потім почнуть платити за кожні 1000 викликів сервісу протягом довгого часу	каналу збуту, фінансування збутових операцій. Обслуговування проданих товарів.		

Отже, було сформовано систему збуту у вигляді щомісячної оплати за певну кількість викликів сервісу, та буде включати користувацьку підтримку. Збут буде проводитися власними силами.

Далі розробляється концепція маркетингових комунікацій (табл. 4.22).

Таблиця 4.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Це інтернет-компанії, які надають перевагу новітнім технологіям	E-mail, письмові звернення до компаній, конференції	Унікальна особливість чат-ботів - емоційність, покращує задоволення клієнтів	Привернути максимум уваги до продукту, змусити компанію спробувати продукт	Користувачі хочуть спілкуватись з живими людьми, компанії хочуть наймати менше людей - ми компроміс.

Отже, було розроблено концепцію маркетингових комунікацій: ключові позиції для позиціонування – емоційність розроблених чат-ботів при спілкуванні, основне завдання рекламного повідомлення - привернути максимум уваги до продукту, змусити компанію спробувати продукт, з фокусуванням на концепцію рекламного звернення – компроміс між потребами компаній та їх користувачів.

4.6 ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі були досліджені основні аспекти виходу на ринок веб-сервісу “Емоційна діалогова система для рекомендації онлайн курсів”. Описаний продукт є доцільний для користувачів, які бажають шукати онлайн курси напряму у власному текстовому мессенджері чи людей, які намагаються покращити рівень володіння англійською мовою.

В рамках розділу було визначено перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару, що є підґрунтям для формування його конкурентоспроможності; обрана технологія реалізації ідеї проекту: для створення нейронної діалогової системи - PyTorch, яка є безкоштовною, зручною та гнучкою для експлуатації; проведений ступеневий аналіз конкуренції на ринку, SWOT аналіз та обґрунтовані фактори конкурентоспроможності. Також було вибрано основні цільові групи: компанії, які потребують автоматизацію відділу підтримки та компанії, які потребують автоматизацію відділу продажів чи прийняття замовлень, адже обидві групи мають схожі вимоги та готові сприйняти продукт. Товар буде мати вигляд чат-боту, який автоматизуватиме користувацькі бізнес-процеси, а саме користувацької підтримки та відділу продажів. Його основні характеристики: емоційність, надійність та найдовший час відповіді. Технологія буде захищена від копіювання за рахунок патенту.

Отже відповідно до проведених досліджень:

- існує можливість ринкової комерціалізації проекту;
- існують перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження не є високими, проект має дві значні

переваги перед конкурентами: кросплатформність, а також емоційність чат-бота;

- необхідно реалізувати нейромережеву діалогову систему на базі фреймворку PyTorch;
- подальша імплементація є доцільною.

ВИСНОВКИ

В результаті виконання даної роботи було здійснено аналіз алгоритмів класифікації емоцій у текстових даних, розроблено нейромережеву архітектуру для контролю емоцій у генеративних діалогових системах та механізм семантичного пошуку онлайн курсів. Крім того, всі дані напрацювання були успішно об’єднані для створення емоційної нейронної діалогової системи для пошуку онлайн курсів.

У роботі висвітлена теоретична база кожного з алгоритмів та методів, а також показані значення метрик, а саме F1-оцінки для класифікації та заплутаності для моделі мови діалогового агента, які висвітлюють ефективність та перевагу того чи іншого підходу відносно інших. Для демонстрації цього були використані відомі набори даних такі як Cornell Movie Dialogs Corpus та Crowdflower's The Emotion in Text, які вважаються класичними вибірками для тестування такого роду алгоритмів. Для рекомендації онлайн курсів був зібраний власний набір даних з загальним описом кожної освітньої програми на відомих платформах масових відкритих онлайн курсів.

Одним із підзавдань роботи було створення класифікатора емоцій, який би міг анотувати діалоговий набір даних для тренування ЕНДС. Проведене дослідження у даному напрямку дозволило обрати серед низки алгоритмів машинного навчання найкращий, яким виявився FastText. Він зміг досягнути F1-оцінку у 0.51, що стало прийнятною точністю для використання його у подальших етапах роботи. Крім того було продемонстровано оптимальні гіперпараметри, підібрані методом сіткового пошуку, для тренування FastText під завданню класифікації емоцій.

Наступним етапом дослідження була побудова архітектури рекурентної нейронної мережі для врахування емоційного забарвлення відповіді у діалозі. Через введення представлень емоцій - додаткового шару у типовій LSTM-Seq2seq мережі, на вхід декодера подаються не лише вектор вхідного слова, а його конкатенація з вектором емоції. Таким чином розроблена архітектура надає можливість системі розуміти концепцію емоції і відповідати у заданому ключі. Експерименти показали, що запропонована модель здатна генерувати відповіді, які підходять не тільки за

змістом, але і за емоцією. При цьому було отримано 6-7% покращення заплутаності мовної моделі діалогової системи порівняно з базовим підходом Seq2seq.

Наступним елементом дослідження стала розробка механізму рекомендації освітніх курсів. У основі отриманого підходу лежить семантична близькість представлень списку курсів зі створеного набору даних до запиту користувача. Крім того, у роботі було продумано як пришвидшити процес пошуку за допомогою групування курсів у категорії. Також наведено рекомендації по підбору параметра системи - порогу семантичної віддаленості - для найбільш ефективної генерації відповідей.

Результатом роботи є готовий набір методів для побудови діалогових систем з певними характеристиками та тематикою. На прикладі характеристики емоційності та тематики онлайн курсів продемонстровано, що досліджені підходи можна ефективно використовувати на практиці.

Так як діалогові системи вцілому набувають все більшого поширення, потенційні застосування даної роботи також розширюються та включають в себе:

1) розробку чат-ботів з активою емоційною складовою, які можуть краще імітувати людське спілкування. Такі чат-боти наприклад можна використовувати для автоматичної користувацької підтримки чи навчання іноземним мовам;

2) використання отриманих результатів для порівняння з новими підходами, методами та алгоритмами для введення емоційності у діалогові системи. Так як це одна з перших робіт у даному напрямку для англійської мови, наступні дослідження можуть відштовхуватись від даного як базового і порівнювати заплутаність їхніх архітектур із зазначеною у роботі;

Отже, конкатенація представлень емоцій з векторами слів на вході до декодера seq2seq рекурентної нейронної мережі може надати діалоговій системі змогу висловлюватись з різними емоційними забарвленнями та зменшити заплутаність мережі при генерації відповідей у діалозі. Також, побудова представлень об’єктів прикладної області(у даній роботі - онлайн курсів) з їх текстового опису надає можливість використати семантичний пошук відповідно до текстового запиту

користувача. Разом дані два компонента дозволяють побудувати гібридну діалогову систему, яка більш ефективно імітує людське спілкування - через емоційну складову, та виконує корисну роботу через рекомендацію об’єктів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Deep Learning for Chatbots [Електронний ресурс] – Режим доступу до ресурсу: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.
2. Salovey P. Emotional intelligence / P. Salovey, J. D. Mayer. // Imagination, cognition and personality. – 1990. – №9. – С. 185–211.
3. Bartneck C. The relationship between emotion models and artificial intelligence / C. Bartneck, M. J. Lyons, M. Saerbeck. // arXiv:1706.09554. – 2017.
4. Vinyals O. A Neural Conversational Model / O. Vinyals, Q. Le. // arXiv:1506.05869. – 2015.
5. Emotional chatting machine: emotional conversation generation with internal and external memory / [H. Zhou, M. Huang, T. Zhang та ін.]. // arXiv:1704.01074. – 2017.
6. A Knowledge-Grounded Neural Conversation Model / [M. Ghazvininejad, C. Brockett, M. Chang та ін.]. // arXiv:1702.01932. – 2017.
7. Google's neural machine translation system: Bridging the gap between human and machine translation / [W. Yonghui, M. Schuster, Z. Chen та ін.]. // arXiv:1609.08144. – 2016.
8. A Diversity-Promoting Objective Function for Neural Conversation Models / [J. Li, M. Galley, C. Brockett та ін.]. // arXiv:1510.03055. – 2016.
9. Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models / [L. Shao, S. Gouws, D. Britz та ін.]. – 2017.
10. A Persona-Based Neural Conversation Model / [J. Li, M. Galley, C. Brockett та ін.]. // arXiv:1603.06155. – 2016.
11. Sentiment Analysis: Emotion in Text [Електронний ресурс] – Режим доступу до ресурсу: https://www.crowdflower.com/data/sentiment-analysis-emotion-text/.
12. Mohammad S. M. # Emotional tweets / Saif M. Mohammad. // Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings

of the Sixth International Workshop on Semantic Evaluation. Association for Comput. – 2012.

13. Mohammad S. M. Using Hashtags to Capture Fine Emotion Categories from Tweets. / S. M. Mohammad, S. Kiritchenko. // Computational Intelligence. – 2015. – №31. – С. 301–326.
14. Emotion, Sentiment, and Stance Labeled Data [Електронний ресурс] – Режим доступу до ресурсу: http://saifmohammad.com/WebPages/SentimentEmotionLabeledData.html.
15. Mohammad S. M. Emotion Intensities in Tweets / S. M. Mohammad, F. Bravo-Marquez. // In Proceedings of the sixth joint conference on lexical and computational semantics (*Sem). – 2017.
16. Multi-emotion detection in user-generated reviews / L.Buitinck, J. Amerongen, E. Tan, M. Rijke. // Proc. 37th European Conference on Information Retrieval (ECIR). – 2015.
17. Strapparava C. SemEval-2007 Task 14: Affective Text / C. Strapparava, R. Mihalcea. // in Proceedings of the 4th International Workshop on the Semantic Evaluations (SemEval 2007). – 2007.
18. Дерево прийняття рішень [Електронний ресурс] - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Дерево_прийняття_рішень.
19. Random forests [Електронний ресурс] - Режим доступу до ресурсу: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.
20. Логістична регресія [Електронний ресурс] - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Logistic_regression.
21. Bag of tricks for efficient text classification / A.Joulin, E. Grave, P. Bojanowski, T. Mikolov. // arXiv:1607.01759. – 2016.
22. Hochreiter S. Long short-term memory / S. Hochreiter, J. Schmidhuber. // Neural computation. – 1997. – №10. – С. 1735–1780.
23. Wiseman S. Sequence-to-sequence learning as beam-search optimization / S. Wiseman, A. M. Rush. // arXiv:1606.02960. – 2016.

24. Williams R. J. A learning algorithm for continually running fully recurrent neural networks / R. J. Williams, D. Zipser. // Neural computation. – 1989. – №1. – С. 270–280.
25. Kingma D. P. Adam: A method for stochastic optimization / D. P. Kingma, J. Ba. // arXiv:1412.6980. – 2014.
26. Danescu-Niculescu-Mizil C. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs / C. Danescu-Niculescu-Mizil, L. Lee. // Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics. Association for Computational Linguistics. – 2011.
27. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems / R.Lowe, N. Pow, I. V. Serban, J. Pineau. // arXiv:1506.08909. – 2015.
28. A neural network approach to context-sensitive generation of conversational responses / [A. Sordoni, M. Galley, M. Auli та ін.]. // arXiv:1506.06714. – 2015.
29. all public Reddit comments, 250gb compressed [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/data/31657.
30. Recommender systems in technology enhanced learning / [N. Manouselis, H. Drachsler, R. Vuorikari та ін.] // Recommender systems handbook / [N. Manouselis, H. Drachsler, R. Vuorikari та ін.]., 2011. – С. 387–415.
31. Efficient Estimation of Word Representations in Vector Space / T.Mikolov, K. Chen, G. Corrado, J. Dean. // In Proceedings of Workshop at ICLR. – 2013.
32. Shlens J. A tutorial on principal component analysis / Jonathon Shlens. // arXiv:1404.1100. – 2014.
33. Maaten L. Visualizing data using t-SNE / L. Maaten, G. Hinton. // Journal of Machine Learning Research. – 2008.
34. Hackeling Gavin. Mastering Machine Learning with scikit-learn / Hackeling Gavin. - Packt Publishing Ltd. - 2014. – С. 1-32.

35. TensorFlow: A System for Large-Scale Machine Learning / [M. Abadi, P. Barham, J. Chen та ін.]. // OSDI. – 2016. – №16.
36. Automatic differentiation in PyTorch / [A. Paszke, S. Gross, S. Chintala та ін.]. // 31st Conference on Neural Information Processing Systems (NIPS 2017). – 2017.
37. Hardware for machine learning: Challenges and opportunities / [V. Sze, Y. Chen, J. Emer та ін.]. // IEEE Custom Integrated Circuits Conference (CICC). – 2017.
38. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation / [C. Liu, R. Lowe, I. V. Serban та ін.]. // arXiv:1603.08023. – 2016.

ДОДАТОК А

Лістинг емоційної нейронної діалогової системи

```
import random

import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.autograd import Variable

from src.models.conversational.model import Seq2seq, BaseRNN, Attention, DecoderRNN, inflate


class EmotionSeq2seq(Seq2seq):

    def forward(self, input_variable, input_lengths=None, target_variable=None, target_emotion=None, teacher_forcing_ratio=0.):
        """Seq2seq forward pass.

        Args:
            input_variable (list): List of token IDs.
            input_lengths (Optional[list(int)]): Lengths of sequences. Defaults to None.
            target_variable (Optional[list]): List of token IDs. Defaults to None.
            target_emotion (Optional[list]): List of emotion IDs. Defaults to None.
            teacher_forcing_ratio (Optional[float]): Teaching forcing ratio. Defaults to 0.

        Returns:
            torch.Tensor(seq_len, batch, vocab_size): Decoding outputs.
            torch.Tensor(num_layers * num_directions, batch, hidden_size): Hidden state.
            dict: {*KEY_LENGTH* : lengths of output sequences, *KEY_SEQUENCE* : predicted token IDs}.

        """
        encoder_outputs, encoder_hidden = self.encoder(input_variable, input_lengths)
        result = self.decoder(inputs=target_variable,
                              emotion_inputs=target_emotion,
                              encoder_hidden=encoder_hidden,
                              encoder_outputs=encoder_outputs,
```

```
                function=self.decode_function,
                teacher_forcing_ratio=teacher_forcing_ratio)
        return result


class EmotionDecoderRNN(BaseRNN):
    """Provides decoding in a seq2seq framework, with a controllable input(emotion).

    Attributes:
        KEY_ATTN_SCORE (str): key used to indicate attention weights in `ret_dict`
        KEY_LENGTH (str): key used to indicate a list representing lengths of output sequences in `ret_dict`
        KEY_SEQUENCE (str): key used to indicate a list of sequences in `ret_dict`

    """

    KEY_ATTN_SCORE = 'attention_score'
    KEY_LENGTH = 'length'
    KEY_SEQUENCE = 'sequence'

    def __init__(self, vocab_size, emotion_vocab_size, max_len, embeddings_dim, emotion_embeddings_dim, hidden_size,
                 sos_id, eos_id, n_layers=1, rnn_cell='gru', bidirectional=False,
                 input_dropout_p=0, dropout_p=0, use_attention=False):
        """Build decoder layers.

        Args:
            vocab_size (int): Size of the vocabulary.
            emotion_vocab_size (int): Size of the emotion vocabulary.
            max_len (int): A maximum allowed length for the sequence to be processed.
            embeddings_dim (int): The size of embeddings vectors.
            emotion_embeddings_dim (int): The size of emotion embedding vectors.
            hidden_size (int): The number of features in the hidden state `h`.
            sos_id (int): Index of the start of sentence symbol.
            eos_id (int): Index of the end of sentence symbol.
            n_layers (Optional[int]): Number of recurrent layers. Defaults to 1.
            rnn_cell (Optional[str]): Type of RNN cell. Defaults to gru.
            bidirectional (Optional[bool]): If the encoder is bidirectional. Defaults to False.
            input_dropout_p (Optional[float]): Dropout probability for the input sequence. Defaults to 0.
```

```
        dropout_p (Optional[float]): Dropout probability for the output sequence. Defaults to 0.
        use_attention(Optional[bool]): Flag indication whether to use attention mechanism or not. Defaults to False.

    """
    super().__init__(vocab_size, max_len, embeddings_dim, hidden_size,
                     input_dropout_p, dropout_p,
                     n_layers, rnn_cell)

    self.bidirectional_encoder = bidirectional
    self.rnn = self.rnn_cell(embeddings_dim + emotion_embeddings_dim, hidden_size, n_layers, batch_first=True,
                             dropout=dropout_p)

    self.output_size = vocab_size
    self.emotion_vocab_size = emotion_vocab_size
    self.emotion_embeddings_dim = emotion_embeddings_dim
    self.max_length = max_len
    self.use_attention = use_attention
    self.eos_id = eos_id
    self.sos_id = sos_id

    self.init_input = None

    self.embedding = nn.Embedding(self.output_size, self.embeddings_dim)
    self.emotion_embedding = nn.Embedding(self.emotion_vocab_size, self.emotion_embeddings_dim)
    if use_attention:
        self.attention = Attention(self.hidden_size)

    self.out = nn.Linear(self.hidden_size, self.output_size)

def forward_step(self, input_var, input_emotion, hidden, encoder_outputs, function):
    batch_size = input_var.size(0)
    output_size = input_var.size(1)
    embedded = self.embedding(input_var)
    embedded = self.input_dropout(embedded)
    embedded_emotion = self.emotion_embedding(input_emotion)
    embedded = torch.cat((embedded, embedded_emotion.view((batch_size, 1, self.emotion_embeddings_dim)).expand((batch_size, output_size,
self.emotion_embeddings_dim))), dim=2)
```

```
    output, hidden = self.rnn(embedded, hidden)

    attn = None
    if self.use_attention:
        output, attn = self.attention(output, encoder_outputs)

    predicted_softmax = function(self.out(output.contiguous().view(-1, self.hidden_size)), dim=1).view(batch_size, output_size,
                                                                                                        -1)
    return predicted_softmax, hidden, attn

def forward(self, inputs=None, emotion_inputs=None, encoder_hidden=None, encoder_outputs=None,
            function=F.log_softmax, teacher_forcing_ratio=0.):
    """Decoder forward pass.

    Args:
        inputs (Optional[torch.Tensor(batch, seq_len, input_size)]): List of token IDs for teacher forcing.
            Defaults to None.
        emotion_inputs (Optional[torch.Tensor(batch, seq_len, input_size)]): Emotion category.
            Defaults to None.
        encoder_hidden (Optional[torch.Tensor(num_layers * num_directions, batch_size, hidden_size)]): Hidden
            state `h` of encoder for the initial hidden state of the decoder. Defaults to None.
        encoder_outputs (Optional[torch.Tensor(batch, seq_len, hidden_size)]): Outputs of the encoder
            for attention mechanism. Defaults to None.
        function (Optional[torch.nn.Module]): Function to generate symbols from RNN hidden state.
            Defaults to `torch.nn.functional.log_softmax`.
        teacher_forcing_ratio (Optional[float]): Teaching forcing ratio. Defaults to 0.

    Returns:
        torch.Tensor(seq_len, batch, vocab_size): Decoding outputs.
        torch.Tensor(num_layers * num_directions, batch, hidden_size): Hidden state.
        dict: {*KEY_LENGTH* : lengths of output sequences, *KEY_SEQUENCE* : predicted token IDs}.
    """
    ret_dict = dict()
    if self.use_attention:
        ret_dict[DecoderRNN.KEY_ATTN_SCORE] = list()

    inputs, batch_size, max_length = self._validate_args(inputs, encoder_hidden, encoder_outputs,
```

```
                                        function)
decoder_hidden = self._init_state(encoder_hidden)

use_teacher_forcing = True if random.random() < teacher_forcing_ratio else False

decoder_outputs = []
sequence_symbols = []
lengths = np.array([max_length] * batch_size)

def decode(step, step_output, step_attn):
    decoder_outputs.append(step_output)
    if self.use_attention:
        ret_dict[DecoderRNN.KEY_ATTN_SCORE].append(step_attn)
    symbols = decoder_outputs[-1].topk(1)[1]
    sequence_symbols.append(symbols)

    eos_batches = symbols.data.eq(self.eos_id)
    if eos_batches.dim() > 0:
        eos_batches = eos_batches.cpu().view(-1).numpy()
        update_idx = ((lengths > step) & eos_batches) != 0
        lengths[update_idx] = len(sequence_symbols)
    return symbols

if use_teacher_forcing:
    decoder_input = inputs[:, :-1]
    decoder_output, decoder_hidden, attn = self.forward_step(decoder_input, emotion_inputs, decoder_hidden, encoder_outputs,
                                            function=function)

    for di in range(decoder_output.size(1)):
        step_output = decoder_output[:, di, :]
        if attn is not None:
            step_attn = attn[:, di, :]
        else:
            step_attn = None
        decode(di, step_output, step_attn)
else:
    decoder_input = inputs[:, 0].unsqueeze(1)
```

```
        for di in range(max_length):
            decoder_output, decoder_hidden, step_attn = self.forward_step(decoder_input, emotion_inputs, decoder_hidden,
                                                           encoder_outputs,
                                                           function=function)
            step_output = decoder_output.squeeze(1)
            symbols = decode(di, step_output, step_attn)
            decoder_input = symbols

    ret_dict[DecoderRNN.KEY_SEQUENCE] = sequence_symbols
    ret_dict[DecoderRNN.KEY_LENGTH] = lengths.tolist()

    return decoder_outputs, decoder_hidden, ret_dict

def _init_state(self, encoder_hidden):
    """Initialize the encoder hidden state."""
    if encoder_hidden is None:
        return None
    if isinstance(encoder_hidden, tuple):
        encoder_hidden = tuple([self._cat_directions(h) for h in encoder_hidden])
    else:
        encoder_hidden = self._cat_directions(encoder_hidden)
    return encoder_hidden

def _cat_directions(self, h):
    """If the encoder is bidirectional, do the following transformation.
        (#directions * #layers, #batch, hidden_size) -> (#layers, #batch, #directions * hidden_size)
    """
    if self.bidirectional_encoder:
        h = torch.cat([h[0:h.size(0):2], h[1:h.size(0):2]], 2)
    return h

def _validate_args(self, inputs, encoder_hidden, encoder_outputs, function):
    if self.use_attention:
        if encoder_outputs is None:
            raise ValueError("Argument encoder_outputs cannot be None when attention is used.")

    # inference batch size
```

```
        if inputs is None and encoder_hidden is None:
            batch_size = 1
        else:
            if inputs is not None:
                batch_size = inputs.size(0)
            else:
                if self.rnn_cell is nn.LSTM:
                    batch_size = encoder_hidden[0].size(1)
                elif self.rnn_cell is nn.GRU:
                    batch_size = encoder_hidden.size(1)

        # set default input and max decoding length
        if inputs is None:
            inputs = Variable(torch.LongTensor([self.sos_id] * batch_size),
                              volatile=True).view(batch_size, 1)
            if torch.cuda.is_available():
                inputs = inputs.cuda()
            max_length = self.max_length
        else:
            max_length = inputs.size(1) - 1  # minus the start of sequence symbol

        return inputs, batch_size, max_length



class EmotionTopKDecoder(torch.nn.Module):
    """Top-K decoding with beam search."""

    def __init__(self, decoder_rnn, k):
        """Builds the decoder layers.

        Args:
            decoder_rnn (EmotionDecoderRNN): An object of EmotionDecoderRNN used for decoding.
            k (int): Size of the beam.

        """
        super().__init__()
        self.rnn = decoder_rnn
```

```
    self.k = k
    self.hidden_size = self.rnn.hidden_size
    self.V = self.rnn.output_size
    self.SOS = self.rnn.sos_id
    self.EOS = self.rnn.eos_id

def forward(self, inputs=None, emotion_inputs=None, encoder_hidden=None, encoder_outputs=None, function=F.log_softmax,
            teacher_forcing_ratio=0, retain_output_probs=True):
    """
    Args:
        inputs (Optional[torch.Tensor(batch, seq_len, input_size)]): List of token IDs for teacher forcing.
            Defaults to None.
        emotion_inputs (Optional[torch.Tensor(batch, 1)]): Response emotion IDs. Defaults to None.
        encoder_hidden (Optional[torch.Tensor(num_layers * num_directions, batch_size, hidden_size)]): Hidden
            state `h` of encoder for the initial hidden state of the decoder. Defaults to None.
        encoder_outputs (Optional[torch.Tensor(batch, seq_len, hidden_size)]): Outputs of the encoder
            for attention mechanism. Defaults to None.
        function (Optional[torch.nn.Module]): Function to generate symbols from RNN hidden state.
            Defaults to `torch.nn.functional.log_softmax`.
        teacher_forcing_ratio (Optional[float]): Probability of teacher forcing. Defaults to 0.
        retain_output_probs (bool): If doing local backpropagation, retain the output layer. Defaults to True.

    Returns:
        torch.Tensor(seq_len, batch, vocab_size): Decoding outputs.
        torch.Tensor(num_layers * num_directions, batch, hidden_size): Hidden state.
        {}: {*length* : lengths of output sequences, *topk_length*: lengths of beam search sequences,
            *sequence* : list of predicted token IDs, *topk_sequence* : list of token IDs from beam search,
            *inputs* : target outputs if provided for decoding}.

    """
    inputs, batch_size, max_length = self.rnn._validate_args(inputs, encoder_hidden, encoder_outputs,
                                                              function)

    pos_index = torch.LongTensor(range(batch_size)) * self.k
    if torch.cuda.is_available():
        pos_index = pos_index.cuda()
    self.pos_index = Variable(pos_index).view(-1, 1)
```

```
# Inflate the initial hidden states to be of size: b*k x h
encoder_hidden = self.rnn._init_state(encoder_hidden)
if encoder_hidden is None:
    hidden = None
else:
    if isinstance(encoder_hidden, tuple):
        hidden = tuple([inflate(h, self.k, 1) for h in encoder_hidden])
    else:
        hidden = inflate(encoder_hidden, self.k, 1)

# ... same idea for encoder_outputs and decoder_outputs
if self.rnn.use_attention:
    inflated_encoder_outputs = inflate(encoder_outputs, self.k, 0)
else:
    inflated_encoder_outputs = None

# Initialize the scores; for the first step,
# ignore the inflated copies to avoid duplicate entries in the top k
sequence_scores = torch.Tensor(batch_size * self.k, 1)
sequence_scores.fill_(-float('Inf'))
sequence_scores.index_fill_(0, torch.LongTensor([i * self.k for i in range(0, batch_size)]), 0.0)
if torch.cuda.is_available():
    sequence_scores = sequence_scores.cuda()
sequence_scores = Variable(sequence_scores)

# Initialize the input vector
input_seq = torch.transpose(torch.LongTensor([[self.SOS] * batch_size * self.k]), 0, 1)
if torch.cuda.is_available():
    input_seq = input_seq.cuda()
input_var = Variable(input_seq)

emotion_inputs = inflate(emotion_inputs, self.k, dim=0)

# Store decisions for backtracking
stored_outputs = list()
stored_scores = list()
```

```
stored_predecessors = list()
stored_emitted_symbols = list()
stored_hidden = list()

for _ in range(0, max_length):

    # Run the RNN one step forward
    log_softmax_output, hidden, _ = self.rnn.forward_step(input_var, emotion_inputs, hidden,
                                          inflated_encoder_outputs, function=function)

    # If doing local backprop (e.g. supervised training), retain the output layer
    if retain_output_probs:
        stored_outputs.append(log_softmax_output)

    # To get the full sequence scores for the new candidates, add the local scores for t_i to the predecessor scores for t_(i-1)
    sequence_scores = inflate(sequence_scores, self.V, 1)
    sequence_scores += log_softmax_output.squeeze(1)
    scores, candidates = sequence_scores.view(batch_size, -1).topk(self.k, dim=1)

    # Reshape input = (bk, 1) and sequence_scores = (bk, 1)
    input_var = (candidates % self.V).view(batch_size * self.k, 1)
    sequence_scores = scores.view(batch_size * self.k, 1)

    # Update fields for next timestep
    predecessors = (candidates / self.V + self.pos_index.expand_as(candidates)).view(batch_size * self.k, 1)
    if isinstance(hidden, tuple):
        hidden = tuple([h.index_select(1, predecessors.squeeze()) for h in hidden])
    else:
        hidden = hidden.index_select(1, predecessors.squeeze())

    # Update sequence scores and erase scores for end-of-sentence symbol so that they aren't expanded
    stored_scores.append(sequence_scores.clone())
    eos_indices = input_var.data.eq(self.EOS)
    if eos_indices.nonzero().dim() > 0:
        sequence_scores.data.masked_fill_(eos_indices, -float('inf'))

    # Cache results for backtracking
```

```
            stored_predecessors.append(predecessors)
            stored_emitted_symbols.append(input_var)
            stored_hidden.append(hidden)

        # Do backtracking to return the optimal values
        output, h_t, h_n, s, l, p = self._backtrack(stored_outputs, stored_hidden,
                                                    stored_predecessors, stored_emitted_symbols,
                                                    stored_scores, batch_size, self.hidden_size)

        # Build return objects
        decoder_outputs = [step[:, 0, :] for step in output]
        if isinstance(h_n, tuple):
            decoder_hidden = tuple([h[:, :, 0, :] for h in h_n])
        else:
            decoder_hidden = h_n[:, :, 0, :]
        metadata = {}
        metadata['inputs'] = inputs
        metadata['output'] = output
        metadata['h_t'] = h_t
        metadata['score'] = s
        metadata['topk_length'] = l
        metadata['topk_sequence'] = p
        metadata['length'] = [seq_len[0] for seq_len in l]
        metadata['sequence'] = [seq[0] for seq in p]
        return decoder_outputs, decoder_hidden, metadata

    def _backtrack(self, nw_output, nw_hidden, predecessors, symbols, scores, b, hidden_size):
        """Backtracks over batch to generate optimal k-sequences.

        Args:
            nw_output [(batch*k, vocab_size)] * sequence_length: A Tensor of outputs from network.
            nw_hidden [(num_layers, batch*k, hidden_size)] * sequence_length: A Tensor of hidden states from network.
            predecessors [(batch*k)] * sequence_length: A Tensor of predecessors.
            symbols [(batch*k)] * sequence_length: A Tensor of predicted tokens.
            scores [(batch*k)] * sequence_length: A Tensor containing sequence scores
                for every token t = [0, ... , seq_len - 1].
            b (int): Size of the batch.
```

```
    hidden_size (int): Size of the hidden state.

Returns:
    output [(batch, k, vocab_size)] * sequence_length: A list of the output probabilities (p_n)
        from the last layer of the RNN, for every n = [0, ... , seq_len - 1].
    h_t [(batch, k, hidden_size)] * sequence_length: A list containing the output features (h_n)
        from the last layer of the RNN, for every n = [0, ... , seq_len - 1].
    h_n(batch, k, hidden_size): A Tensor containing the last hidden state for all top-k sequences.
    score [batch, k]: A list containing the final scores for all top-k sequences.
    length [batch, k]: A list specifying the length of each sequence in the top-k candidates.
    p (batch, k, sequence_len): A Tensor containing predicted sequence.
"""

lstm = isinstance(nw_hidden[0], tuple)

# initialize return variables given different types
output = list()
h_t = list()
p = list()
# Placeholder for last hidden state of top-k sequences.
# If a (top-k) sequence ends early in decoding, `h_n` contains
# its hidden state when it sees EOS.  Otherwise, `h_n` contains
# the last hidden state of decoding.
if lstm:
    state_size = nw_hidden[0][0].size()
    h_n = tuple([torch.zeros(state_size), torch.zeros(state_size)])
else:
    h_n = torch.zeros(nw_hidden[0].size())
l = [[self.rnn.max_length] * self.k for _ in range(b)]  # Placeholder for lengths of top-k sequences
# Similar to `h_n`

# the last step output of the beams are not sorted
# thus they are sorted here
sorted_score, sorted_idx = scores[-1].view(b, self.k).topk(self.k)
# initialize the sequence scores with the sorted last step beam scores
s = sorted_score.clone()
```

```
batch_eos_found = [0] * b  # the number of EOS found
# in the backward loop below for each batch

t = self.rnn.max_length - 1
# initialize the back pointer with the sorted order of the last step beams.
# add self.pos_index for indexing variable with b*k as the first dimension.
t_predecessors = (sorted_idx + self.pos_index.expand_as(sorted_idx)).view(b * self.k)
while t >= 0:
    # Re-order the variables with the back pointer
    current_output = nw_output[t].index_select(0, t_predecessors)
    if lstm:
        current_hidden = tuple([h.index_select(1, t_predecessors) for h in nw_hidden[t]])
    else:
        current_hidden = nw_hidden[t].index_select(1, t_predecessors)
    current_symbol = symbols[t].index_select(0, t_predecessors)
    # Re-order the back pointer of the previous step with the back pointer of
    # the current step
    t_predecessors = predecessors[t].index_select(0, t_predecessors).squeeze()

    # This tricky block handles dropped sequences that see EOS earlier.
    # The basic idea is summarized below:
    #
    #   Terms:
    #       Ended sequences = sequences that see EOS early and dropped
    #       Survived sequences = sequences in the last step of the beams
    #
    #       Although the ended sequences are dropped during decoding,
    #   their generated symbols and complete backtracking information are still
    #   in the backtracking variables.
    #   For each batch, everytime we see an EOS in the backtracking process,
    #       1. If there is survived sequences in the return variables, replace
    #       the one with the lowest survived sequence score with the new ended
    #       sequences
    #       2. Otherwise, replace the ended sequence with the lowest sequence
    #       score with the new ended sequence
    #
```

```
eos_indices = symbols[t].data.squeeze(1).eq(self.EOS).nonzero()
if eos_indices.dim() > 0:
    for i in range(eos_indices.size(0) - 1, -1, -1):
        # Indices of the EOS symbol for both variables
        # with b*k as the first dimension, and b, k for
        # the first two dimensions
        idx = eos_indices[i]
        b_idx = int(idx[0] / self.k)
        # The indices of the replacing position
        # according to the replacement strategy noted above
        res_k_idx = self.k - (batch_eos_found[b_idx] % self.k) - 1
        batch_eos_found[b_idx] += 1
        res_idx = b_idx * self.k + res_k_idx

        # Replace the old information in return variables
        # with the new ended sequence information
        t_predecessors[res_idx] = predecessors[t][idx[0]]
        current_output[res_idx, :] = nw_output[t][idx[0], :]
        if lstm:
            current_hidden[0][:, res_idx, :] = nw_hidden[t][0][:, idx[0], :]
            current_hidden[1][:, res_idx, :] = nw_hidden[t][1][:, idx[0], :]
            h_n[0][:, res_idx, :] = nw_hidden[t][0][:, idx[0], :].data
            h_n[1][:, res_idx, :] = nw_hidden[t][1][:, idx[0], :].data
        else:
            current_hidden[:, res_idx, :] = nw_hidden[t][:, idx[0], :]
            h_n[:, res_idx, :] = nw_hidden[t][:, idx[0], :].data
        current_symbol[res_idx, :] = symbols[t][idx[0]]
        s[b_idx, res_k_idx] = scores[t][idx[0]].data[0]
        l[b_idx][res_k_idx] = t + 1

# record the back tracked results
output.append(current_output)
h_t.append(current_hidden)
p.append(current_symbol)

t -= 1
```

```
    # Sort and re-order again as the added ended sequences may change
    # the order (very unlikely)
    s, re_sorted_idx = s.topk(self.k)
    for b_idx in range(b):
        l[b_idx] = [l[b_idx][k_idx.data[0]] for k_idx in re_sorted_idx[b_idx, :]]

    re_sorted_idx = (re_sorted_idx + self.pos_index.expand_as(re_sorted_idx)).view(b * self.k)

    # Reverse the sequences and re-order at the same time
    # It is reversed because the backtracking happens in reverse time order
    output = [step.index_select(0, re_sorted_idx).view(b, self.k, -1) for step in reversed(output)]
    p = [step.index_select(0, re_sorted_idx).view(b, self.k, -1) for step in reversed(p)]
    if lstm:
        h_t = [tuple([h.index_select(1, re_sorted_idx).view(-1, b, self.k, hidden_size) for h in step]) for step in
               reversed(h_t)]
        if torch.cuda.is_available():
            h_n = tuple([h.index_select(1, re_sorted_idx.data.cpu()).view(-1, b, self.k, hidden_size) for h in h_n])
        else:
            h_n = tuple([h.index_select(1, re_sorted_idx.data).view(-1, b, self.k, hidden_size) for h in h_n])
    else:
        h_t = [step.index_select(1, re_sorted_idx).view(-1, b, self.k, hidden_size) for step in reversed(h_t)]
        if torch.cuda.is_available():
            h_n = h_n.index_select(1, re_sorted_idx.data.cpu()).view(-1, b, self.k, hidden_size)
        else:
            h_n = h_n.index_select(1, re_sorted_idx.data).view(-1, b, self.k, hidden_size)
    s = s.data

    return output, h_t, h_n, s, l, p

def _mask_symbol_scores(self, score, idx, masking_score=-float('inf')):
    score[idx] = masking_score

def _mask(self, tensor, idx, dim=0, masking_score=-float('inf')):
    if len(idx.size()) > 0:
        indices = idx[:, 0]
        tensor.index_fill_(dim, indices, masking_score)
```